НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)

“___” червня 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**

на тему: Фільтрування акустичного сигналу засобами штучного інтелекту

Виконав: студент IV курсу, групи КВ-51
(шифр групи)

Мартиновець Микола Васильович
(прізвище, ім’я, по батькові) (підпис)

Керівник доц. каф. СПСКС, к.т.н Замятін Д.С
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

ЗАВДАННЯ
на дипломний проект студента
Мартиновця Миколи Васильовича
(прізвище, ім’я, по батькові)

1. Тема проекту

   Фільтрування акустичного сигналу засобами штучного інтелекту, керівник проекту Замятін Денис Станіславович к.т.н. доцент, затверджені наказом по університету від «22» травня 2019 р. №1330-С

2. Термін подання студентом проекту «__» червня 2019 р.

3. Вихідні дані до проекту див. Технічне завдання

4. Зміст пояснювальної записки

   - Аналіз існуючих рішень цифрової обробки акустичних сигналів та обґрунтування теми дипломного проекту
   - Опис алгоритмів фільтрування акустичного сигналу
   - Система видалення шуму з аудіофайлу
   - Тестування реалізації

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- Високорівнева структура більшості систем фільтрації шуму. Структурна схема
- Алгоритм розробленної нейронної мережі. Схема алгоритму
- Алгоритм розбору залишкових стеків. Схема алгоритму
- Алгоритм розбору залишкових блоків. Схема алгоритму.

6. Консультанти розділів проекту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М. доцент		

7. Дата видачі завдання «__» _____ 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вивчення літератури за тематикою проекту	15.04.2019	
2	Розроблення та узгодження технічного завдання	30.04.2019	
3	Аналіз існуючих рішень	05.05.2019	
4	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019	
5	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019	
6	Підготовка графічної частини дипломного проекту	20.05.2019	
7	Оформлення документації дипломного проекту	25.05.2019	
8	Попередній огляд матеріалів диплому на кафедрі	30.05.2019	

Студент _____ Мартиновець М.В
(підпис) (ініціали, прізвище)

Керівник проекту _____ Замятін Д.С
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту.

АНОТАЦІЯ

Об'єктом розробки є система фільтрування акустичного сигналу засобами штучного інтелекту в реальному часі, побудована методом цифрової обробки сигналів з використанням нейронних мереж глибинного навчання.

Розроблена система дозволяє завантажувати аудіозаписи, аналізувати їх, та фільтрувати за заданими ознаками. Всі описані сервіси було розроблено мовою програмування Python. Незважаючи на те, що задача адаптивного подолання шуму є добре вивченою областю в сфері цифрової обробки сигналів, вона залишається в значній мірі залежною від тонких налаштувань алгоритмів та параметрів оцінки. У цій роботі продемонстрований гібридний підхід для очистки вхідного мовного потоку від нерелевантної аудіо інформації. Рекурентна нейронна мережа з чотирма прихованими шарами використовується щоб оцінити ідеальні критичні коефіцієнти підсилення, тоді як більш традиційний фільтр тону знижує шум між гармоніками тону. Данний метод дозволяє досягати значно вищої якості результатів, ніж традиційне застосування мінімальної середньої квадратичної похибки спектральної оцінки, зберігаючи при цьому невеликий розмір нейронної мережі для роботи в реальному часі при частоті вхідного сигналу 48 кГц на пристроях з CPU малої потужності та вискої енергоефективності.

Ключові слова: цифрова обробка сигналів, видалення шуму, рекурентна нейронна мережа, CPU, глибинне навчання.

Abstract

The subject of development is the system of acoustic signal filtration with means of artificial intelligence in real time, built on a digital signal processing system using deep learning neural network.

The developed system allows to upload audio files, analyze them, and also filter them by the specified attributes. All of the described services were developed in the language of the Python program. Despite the fact that the problem of adaptive noise suppression is a well-studied area of digital signaling, it remains largely dependent on fine-tuning of algorithms and parameters. This project demonstrates a hybrid approach to clearing the incoming speech stream from irrelevant audio information. A recurrent neural network with four hidden layers to evaluate ideal critical gain, while a more conventional filter eliminates noise between harmonic tones. This method allows for achieving more higher quality results than the traditional application of minimum mean square spectral speculation, which is provided with a small network size for real-time operation at the frequency of the input signal.

Keywords: digital signal processing, noise removal, recurrent neural network, processor, depth of training.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045490.002 ТЗ	Фільтрування акустичного	4		
			сигналу засобами			
			штучного інтелекту			
			Технічне завдання			
	А4	ІАЛЦ.045490.003 ТП	Фільтрування акустичного	2		
			сигналу засобами			
			штучного інтелекту			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045490.004 ПЗ	Фільтрування акустичного	50		
			сигналу засобами			
			штучного інтелекту			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.001 ОА		
Розробив		Мартиновець М.В			Фільтрування акустичного сигналу засобами штучного інтелекту Опис альбому	Літ.	Аркуш 1 / Аркушів 3
Перевірив		Замятін Д.В.					
Консульт.						НТУУ «КПІ ім. І.Сікорського» Кафедра СПіСКС Група КВ-51	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.005 Д1	Фільтрування акустичного	1		
			сигналу засобами			
			штучного інтелекту			
			Високорівнева структура			
			більшості схем фільтрації			
			шуму			
	А4	ІАЛЦ.045490.006 Д2	Фільтрування акустичного	1		
			сигналу засобами			
			штучного інтелекту			
			Алгоритм розробленої			
			нейронної мережі			
	А4	ІАЛЦ.045490.007 Д3	Фільтрування акустичного	1		
			сигналу засобами			
			штучного інтелекту			
			Алгоритм розбору			
			залишкових стеків			
	А4	ІАЛЦ.045490.008 Д4	Фільтрування акустичного	1		
			сигналу засобами			
			штучного інтелекту			
			Алгоритм розбору			
			залишкових блоків			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ 2

2. ПІДСТАВА ДЛЯ РОЗРОБКИ 2

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ 2

4. ДЖЕРЕЛА РОЗРОБКИ 2

5. ТЕХНІЧНІ ВИМОГИ 2

5.1. Вимоги до програмного продукту, що розробляється 2

5.2. Вимоги до апаратного забезпечення 3

5.3. Вимоги до програмного та апаратного забезпечення користувача 3

6. ЕТАПИ РОЗРОБКИ 4

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Фільтрування акустичного сигналу засобами штучного інтелекту».

Галузь застосування: організація обробки вхідної сенсорної інформації

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання дипломного проекту першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення системи подолання мушу для покращення чистоти аудіопотоку в реальному часі, використовуючи гібридний підхід цифрової обровки сигналів та методів глибинного начання.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з будь якою операційною системою (IOS,Android,WP);
- можливість управління процесом обробки заявок;
- можливість управління заявками;

- можливість додавати нові заявки;
- можливість редагування заявок;
- наявність зручної системи сповіщень ініціатора та виконавця;
- наявність системи контакту з ініціатором;
- прив'язки файлів ініціатора та виконавця до заявок;
- зберігання заявок;
- аналітичні можливості.

5.2. Вимоги до апаратного забезпечення

- Процесор: 2,4-ядерний, MediaTek, Snapdragon, Kirin, Intel Atom;
- Оперативна пам’ять: 2 Гб;
- Наявність доступу до мережі Internet (GPRS, EDGE, 3G, 4G);

5.3. Вимоги до програмного та апаратного забезпечення користувача

- Операційна система Windows Phone 8, Android, iOS;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	30.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ДП.045490.004 ПЗ	Фільтрування акустичного	50		
			сигналу засобами			
			штучного інтелекту			
			Пояснювальна записка			
	А4	ІАЛЦ.045490.005 Д1	Фільтрування акустичного	1		
			сигналу засобами			
			штучного інтелекту			
			Високорівнева структура			
			більшості схем фільтрації			
			шуму			
	А4	ІАЛЦ.045490.006 Д2	Фільтрування акустичного	1		
			сигналу засобами			
			штучного інтелекту			
			Алгоритм розробленої			
			нейронної мережі			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045490.003 ТП		
Розробив		Мартиновець М.В.			Програмна система управління інформаційними потоками в медійній рекламі Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Замятін Д.С.					
Консульт.						НТУУ «КПІ ім. І.Сікорського» Кафедра СПіСКС Група КВ-51	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045490.007 Д3	Фільтрування акустичного	1		
			сигналу засобами			
			штучного інтелекту			
			Алгоритм розбору			
			залишкових стеків			
	А4	ІАЛЦ.045490.008 Д4	Фільтрування акустичного	1		
			сигналу засобами			
			штучного інтелекту			
			Алгоритм розбору			
			залишкових блоків			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

					ІАЛЦ.045490.003 ТП	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		2

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ ____ 3

ВСТУП _____ 4

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ЦИФРОВОЇ ОБРОБКИ АКУСТИЧНИХ СИГНАЛІВ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ _____ 5

1.1. Аналіз актуальності поставленої задачі _____ 5

1.2. Аналіз існуючих рішень для розпізнавання та фільтрування вхідного акустичного сигналу в реальному часі _____ 7

1.3. Формалізація постановки задачі для долідження _____ 16

2. ОПИС АЛГОРИТМІВ ФІЛЬТРУВАННЯ АКУСТИЧНОГО СИГНАЛУ _____ 18

2.1. Моделювання акустичного сигналу _____ 18

2.2. Архітектура глибинного навчання _____ 22

3. СИСТЕМА ВИДАЛЕННЯ ШУМУ З АУДІОФАЙЛУ _____ 27

4. ТЕСТУВАННЯ РЕАЛІЗАЦІЇ _____ 45

ВИСНОВКИ_____ 48

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ_____ 49

ДОДАТКИ

Додаток 1. Копії графічного матеріалів

- ІАЛЦ.045490.005 Д1. Високорівнева структура більшості систем фільтрації шуму. Схема структурна;
- ІАЛЦ.045490.006 Д2. Алгоритм розробленної нейронної мережі. Схема алгоритму;
- ІАЛЦ.045490.007 Д3. Алгоритм розбору залишкових стеків. Схема алгоритму;
- ІАЛЦ.045490.008 Д4. Алгоритм розбору залишкових блоків. Схема алгоритму.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

API – прикладний програмний інтерфейс

BSS – сліпий метод поділу джерел

BFCC – Bark-frequency cepstral coefficients

CNN – Convolutional Neural Network

CTC– Connectionist temporal classification

GRU – Gated Recurrent Unit

DNN – глибинна нейронна мережа

MCEC – алгоритм багатоканального ехоподавлення

MCWF – мультиканальний Wiener фільтр

ML – машинне навчання

Pandas – програмна бібліотека для аналізу та обробки данних

PESQ – Перцептивна оцінка якості мовлення

RES – залишковий подавлювач шуму

SPP – Вхідна ймовірність присутності мовлення

TPU – Tensor Processing Unit

ВСТУП

Технології фільтрування акустичних сигналів є предметом інтересу принаймні з 70-х років минулого століття. Незважаючи на значне поліпшення якості досліджень та методів розробки, структура високого рівня більшості систем залишається однаковою. Техніка спектральної оцінки спирається на спектральну оцінку шуму, яка в той керується детектором голосової активності.

Кожен з 3-х компонентів вимагає точних оцінок та їх сумісну роботу достатньо важко налаштувати. Ось чому останні досягнення в глибинних методах навчання є пріоритетними при обробці акустичних сигналів.

Методи глибинного навчання вже використовуються для фільтрації аудіосигналів, дозволяють реалізовувати видалення нерелеватної аудіо інформації за заданими ознаками. Багато з запропонованих підходів орієнтовані на програми автоматичного розпізнавання мови, де низька затримка та обчислювальна потужність не є вагомими факторами. Крім того, у багатьох випадках великі розміри нейронної мережі ускладнюють реалізацію роботи систем в реального часу без вбудованої GPU. Тому, для досягнення поставленої задачі, в ході цієї розробки був використаний гібридний підхід, який задіює одночасно як добре відомі методи цифрової оброки сигналів так і глибинне навчання для заміни компонент, які потребують детальних налаштувань в звичайних системах.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ЦИФРОВОЇ ОБРОБКИ АКУСТИЧНИХ СИГНАЛІВ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Аналіз актуальності поставленої задачі

У наш час, машинне навчання користується безпрецедентним всплеском популярності серед додатків, які вирішують проблеми сучасного суспільства і дозволяють автоматизувати роботу в різних областях. У першу чергу, це пов'язано з вибухом доступності даних, значним поліпшенням методів машинного навчання та ростом обчислювальних потужностей.

Машинне навчання дозволяє проводити дослідження данних та утворювати нові знання. Це виходить за рамки простого навчання: використання і вдосконалення знань з часом та досвідом. Фактично, мета ML полягає у виявленні та використанні прихованих шаблонів в об’ємі данних призначених для “навчання”. Вивчені структури використовуються для аналізу невідомих даних, такі, що можуть бути поєданими або зіставленими з відомими групами. Це призводить до зміни традиційної парадигми програмування, де програми написані для автоматизації завдань. ML створює програму (модель), яка відповідає даним.

Останнім часом ML користується відновленим інтересом. Ранні методи ML були жорсткими та нездатними переносити будь-які відхилення від навчальних даних. Нещодавні досягнення в галузі розробки машинного навчання зробили ці прийоми гнучкими і стійкими в застосуванні до різних сценаріїв реального світу, для задач різної складності. Наприклад, ML у сфері охорони здоров'я значно покращила сфери медичної візуалізації та автоматизованої діагностики. Зазвичай, ми часто використовуємо технологічні інструменти, які засновані на ML. Наприклад, пошукові системи широко використовують ML для нетривіальних завдань, таких як пропозиції запитів, виправлення правопису, веб-індексування та ранжування сторінок. Очевидно,

коли ми сподіваємося на автоматизацію більшої кількості аспектів нашого життя, починаючи від домашньої автоматизації до автономних транспортних засобів, методи ML стануть все більш важливим аспектом у різних системах, які допомагають у прийнятті рішень, аналізі та автоматизації.

Існують багато областей в полі котрих, були зроблені технологічні прориви задвяки технологіям глибинного навчання. Одне з таких напрямків, в якому глибинне навчання має потенціал для вирішення – обробка аудіопотоку, розпізнавання та синтез мови. Розглянемо задачі з обробки аудіо інформації, в контексті яких прихід методів машинного навчаня став ключовим для їх вирішення.

Аудіо класифікація є фундаментальною проблемою в області обробки звуку. Завдання полягає в тому, щоб вилучити функції з аудіофайлу, а потім визначити, до якого класу належить аудіо. Багато корисних додатків, що стосуються класифікації звуку, можна знайти в дикій природі - наприклад, класифікація жанру, розпізнавання інструментів та ідентифікація виконавця.

Це завдання також є найбільш вивченою темою в обробці аудіофайлів. Протягом останнього року в цій галузі було опубліковано велику кількість статей. Загальний підхід до вирішення завдання класифікації звуку полягає в тому, щоб попередньо обробити аудіовходи для вилучення корисних функцій, а потім застосувати на ньому алгоритм класифікації. Наприклад, у нашому прикладі нижче наводиться 5 секундний уривок звуку, і завдання полягає в тому, щоб визначити, до якого класу воно відноситься - чи то лай собаки або звук буріння. Як згадувалося в статті, підхід до вирішення цього полягає в тому, щоб витягти звукову функцію під назвою MFCC, а потім передати її через нейронну мережу, щоб отримати відповідний клас.

Метою звукової ідентифікація є визначення цифрового «резюме» звуку. Це робиться для ідентифікації звуку з аудіо-зразка. Shazam є прекрасним прикладом застосування звукової ідентифікації. Він розпізнає музику на основі перших двох-п'яти секунд пісні. Тим не менш, все ще існують ситуації,

коли система виходить з ладу, особливо там, де є високий рівень фонового шуму. Щоб вирішити цю проблему, підхід може полягати в тому, щоб представити звук іншим способом, щоб його легко розшифрувати. Потім ми можемо з'ясувати закономірності, які відрізняють звук від фонового шуму. У наведеному нижче прикладі автор перетворює сировину аудіосигнали на спектрограми, а потім використовує алгоритми виявлення піків і відбитків пальців для визначення відбитків цього аудіофайлу.

Позначення музики - це більш складна версія класифікації звуку. Тут можливо мати декілька класів, до яких може належати кожний аудіозапис, так само як і проблема класифікації з декількома мітками. Потенційною можливістю застосування цього завдання може бути створення метаданих для аудіо, щоб його можна було шукати пізніше. Глибинне навчання допомогло вирішити це завдання певною мірою, що можна побачити в прикладі нижче.

Як видно з більшості завдань, перший крок завжди полягає у витяганні функцій з аудіо-зразка. Потім відбуваєтсья сортування відповідно до нюансів звуку (наприклад, якщо аудіозапис містить більше інструментального шуму, ніж голос співака, тег може бути "інструментальним"). Це можна зробити або шляхом машинного навчання, або методами глибокого навчання.

1.2. Аналіз існуючих рішень для розпізнавання та фільтрування вхідного акустичного сигналу в реальному часі

Apple HomePod - це домашній динамік з можливостями розумного будинку, включаючи голосового асистента Siri радіусу далекого реагування, якого користувач може контролювати за допомогою мови навіть під час відтворення музики. Siri на HomePod призначений для роботи в складних сценаріях використання, таких як:

- Під час гучного відтворення музики
- Коли диктор знаходиться далеко від HomePod

- Коли присутні активні джерела звуку в кімнаті, такі як телевізор або побутова техніка

За всіх цих умов Siri на HomePod має відповідати і правильно розпізнавати команди. Багатоканальна система обробки сигналів HomePod грунтується з акцентом на таких підходах:

- Багатоканальна фільтрація на основі маски з використанням глибокого навчання для видалення відбиття звукових коливань від перешкод, коли вони сприймаються роздільно від первісних коливань та фонового шуму
- Навчання без нагляду для відокремлення одночасних джерел звуку та вибору потоку на основі тригер-фрази для усунення заважаючої мови

Система використовує шість мікрофонів і безперервно керує багатоканальною обробкою сигналів на чіпі Apple A8, у тому числі, коли Homepod працює в режимі мінімального енергоспоживання з метою економії енергії. Багатоканальна фільтрація постійно адаптується до мінливих умов шуму і рухомих дикторів.

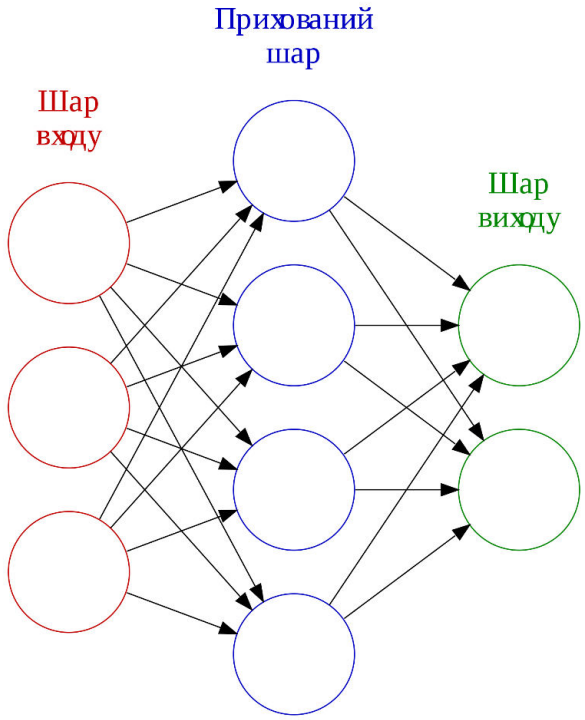


Рисунок 1.1 - Архітектура моделі нейронної мережі

Інші сучасні системи використовують багатомікрофонну обробку для поліпшення мовлення, але зазвичай фокусуються тільки на підмножині таких задач, як фільтрації [1] і видалення шуму [2]. Для придушення небажаних компонентів сигналу система покращення мовлення повинна вивчати бажані і небажані характеристики сигналу, використовуючи методи керування:

- Навчання з наглядом
- Навчання без нагляду

Останнім часом продуктивність вдосконалення мовлення значно покращилася завдяки глибокому навчанню. У роботі [2], наприклад, найбільш ефективні методи вивчають ймовірність присутності мови за допомогою глибокої нейронної мережі (DNN), яка керує фільтрами багатоканального придушення шуму. Тим не менш, ці системи зазвичай побудовані за умови, що повне висловлювання (триггер-фраза), доступне під час виконання і те, що система виконує пакетну обробку щоб скористатися всіма зразками мовлення під час голосової команди. Ця установка збільшує затримку та виключає можливість покращення розпізнавання мовлення для режимів постійного прослуховування на пристроях домашнього помічника, включаючи функціональні можливості виявлення фрази та кінцевої точки. Припущення щодо систем удосконалення пакетного розпізнавання є нереалістичними для HomePod, оскільки акустичні умови непередбачувані, а початкові та кінцеві точки голосових команд недоступні заздалегідь.

Розпізнавання мови у дальній області стає більш складним, коли інший активний диктор, як людина або телевізор, присутній в одній кімнаті з цільовою мовою. У цьому сценарії розпізнавання голосового тригера, декодування мови і кінцева точка може бути істотно погіршена, якщо голосова команда не відокремлена від компонентів мови, що заважають. Традиційно дослідники займаються розділенням джерел мовлення за допомогою методів без нагляду, таких як незалежний компонентний аналіз та кластеризація [4], або глибоке навчання [5, 6]. Ці методи дозволяють покращити автоматичне розпізнавання

мови в додатках конференц-зв'язку або на партіях синтетичних мовних сумішей, де кожен мовний сигнал витягується і транскрибується [6, 7]. На жаль, зручність використання цих пакетних технологій у голосових командних інтерфейсах з великим радіусом дії дуже обмежена. Крім того, вплив поділу джерел на розпізнавання голосового тригера, наприклад, що використовується з "Hey Siri", ніколи раніше не досліджувався. Нарешті, надзвичайно важливо розділяти суміші вхідних конкуруючих сигналів в Інтернеті, щоб уникнути затримок, вибрати і декодувати тільки цільовий потік, що містить голосову команду.

Модель багатоканального сигналу далекого поля може бути записана в частотній області як:

$$y_k[n] = \sum_{i=1}^{N_x} H_k[l]x_k[n-l] + \sum_{p=1}^{N_s} G_k[p]S_k[n-p] + v_k[n] \text{ , де}$$

- n - індекс часу
- k - індекс частоти
- l, p - показники часу імпульсного відгуку
- y_k – це мікрофонні сигнали(M), що містять ехо, реверберацію, шум і конкуруючих дикторів
- x_k - N_x каналів відтворюваних сигналів
- s_k - джерела мови N_s
- v_k - компоненти фонового шуму, захоплені M
- H_k - багатоканальні імпульсні відповіді від гучномовців до мікрофонів, всього MN_x відповідей
- G_k - багатоканальні імпульсні відповіді від джерел мовлення на мікрофони, загальна кількість відповідей MN_s

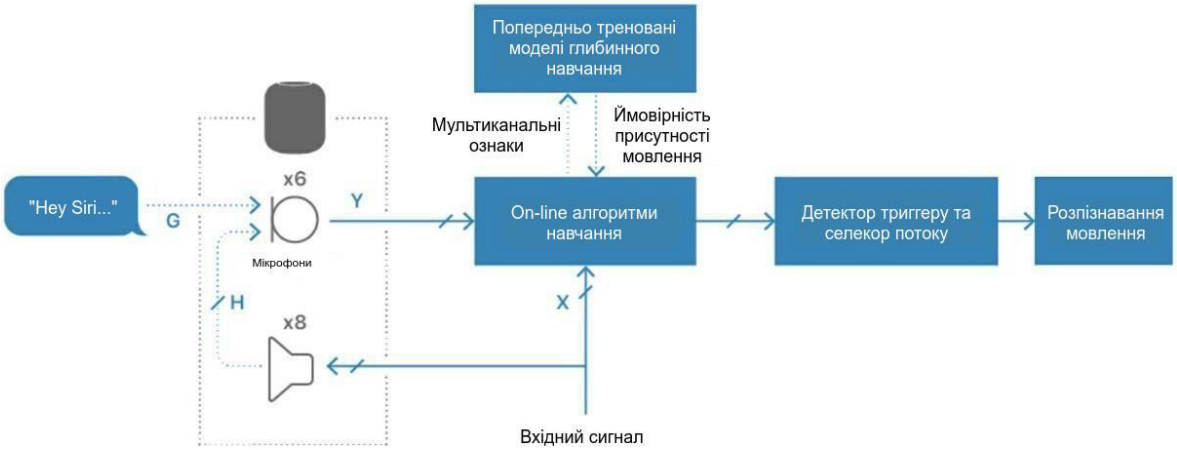


Рисунок 1.2 – Структурна схема багатоканальної мережі обробки сигналів

Через невелику відстань гучномовців до мікрофонів на HomePod, сигнал відтворення може бути значно голоснішим, ніж голосова команда користувача на позиціях мікрофона, особливо коли диктор віддаляється від пристрою. Фактично, ехо-сигнали можуть бути на 30-40 дБ голосніше, ніж мовні сигнали користувача, в результаті чого тригерна фраза не може бути виявлена на мікрофонах під час гучного відтворення музики. Siri на HomePod реалізує алгоритм багатоканального ехоподавлення (MCEC), який використовує набір лінійних адаптивних фільтрів для моделювання декількох акустичних шляхів між гучномовцями і мікрофонами для скасування акустичного зв'язку. На практиці, дві проблеми обмежують MCEC повністю вилучати власний сигнал відтворення з мікрофонів:

- Нелінійний зв'язок. У зв'язку з нелінійністю гучномовця і механічними вібраціями пристроїв, весь сигнал відтворення не захоплюється моделлю лінійної системи [8].
- Невластиві рішення. Коли багатоканальний контент відтворюється з масиву динаміків, рішення може бути не єдиним [9].

Отже, для видалення залишкового вмісту відтворення після МСЕС необхідний залишковий глушник (RES). У наступному розділі я розгляну підхід глубокого навчання до вивчення залишкового придушення відлуння.

При гучному відтворенні МСЕС, зазвичай, незабезпечує достатнє придушення exo для успішного виявлення фрази тригера, що призводить до залишкових рівнів відлуння від 10 до 20 дБ вище мови диктора.

RES розроблений для придушення нелінійних компонентів exo-сигналу, які не моделюються лінійними МСЕС. RES також пом'якшує залишковий лінійний exo, особливо при наявності подвійних розмов і змін шляху відлуння.

Типові підходи придушення залишкового exo-сигналу, керовані даними [10, 11], витягують вхідні функції з опорних і exo-скасованих сигналів та використовують мережу для застосування придушення безпосередньо до exo-скасованого сигналу. У нашому підході до RES, DNN приймає кілька функцій введення і виводить оцінку маски мовної діяльності, яка використовується як вхідна ймовірність присутності мовлення (SPP) для багатоканального Wiener фільтра (MCWF). Вхідні функції витягуються з exo-скасованих сигналів разом з лінійною оцінкою exo-сигналу, що надається МСЕС.

Підхід подалення exo на основі маски має ряд переваг:

- Оскільки глибока нейронна мережа навчалася на фактичних exo-записах, вона навчилася придушувати залишкові exo-сигнали, що виникають внаслідок нелінійності гучномовців і механічних коливань, характерних для HomePod. Ці викривлення нелегко відстежувати і пригнічувати за допомогою традиційних статистичних підходів.
- SPP, передбачений DNN, керує MCWF, щоб направити нульові значення, які значно зменшують залишкове відлуння, отримуючи в результаті низьке спотворення мовлення.

Коли джерело мовлення віддаляється від мікрофонів, багаторазові відображення з кімнати створюють хвости реверберації, які знижують якість і

зрозумілість цільової мови. Сигнал, захоплений мікрофоном, може характеризуватися прямим звуком (мовлення без будь-якого відображення), раннім відбиттям і пізньою реверберацією. Пізня реверберація може сильно погіршити продуктивність розпізнавання мовлення [12 , 13]. Siri на HomePod безперервно відстежує характеристики приміщення та видаляє пізню реверберацію, зберігаючи при цьому в мікрофонних сигналах компоненти прямого і раннього відображення.

На додаток до реверберації, далеке мовлення зазвичай забруднене шумом від побутової техніки, зовнішніх звуків, що надходять через вікна, і різних інших джерел шуму. MCWF на масковій основі є потужним інструментом для розпізнавання мовного шуму в далеких полях у пакетних умовах [3]. На відміну від традиційних одноканальних підходів, MCWF є залежним від сигналу, який може направляти нулі до просторово локалізованих джерел шуму без спотворення цільової мови. Це можливо, коли SPP оцінюється правильно, використовуючи або статистичні моделі або вивчені за допомогою DNN. При обробці в пакеті, цей підхід передбачає, що джерела шуму та мовлення не рухаються під час активних сегментів мовлення.

Ці найсучасніші методи вдосконалення мовлення створюють фіксований фільтр для кожного висловлювання, використовуючи всі сукупні оцінки мови та шуму для підвищення мови перед подачею до розпізнавача. DNN, навчений розпізнавати спектральні характеристики мови і шуму, може ще більше зменшити шум для проблеми оцінки маски. Тим не менш, для боротьби з постійно мінливим акустичним середовищем, вимагаються онлайн-системи зменшення шуму, яка може відстежувати екологічний шум з низькою затримкою. Apple побудували онлайн MCWF, який оцінює статистику мови та шуму, використовуючи тільки поточні та минулі мікрофонні сигнали. Крім того, на серверах розгорнуто DNN, який прогнозує SPP і керує MCWF, щоб направити нульові шляхи до джерел шуму, що заважають.

DNN був навчений на внутрішньо зібраних даних, використовуючи як дифузні, так і спрямовані, як безперервні так і дискретні шуми, які статистично важко моделювати. Синтетично були змішані записи з мовою і записи тільки з шумом, щоб створити записи з перешкодами для мовлення. Розраховані вхідні функції для DNN з розірваного сигналу і оцінки реверберації. Вихідними функціями є маска мовної діяльності, розрахована з аудіо сигналу мови і суміші мовлення та шуму.

Порушення цільової мови іншими втручаючими спікерами є складним завданням для розпізнавання. Сліпий метод поділу джерел (BSS) - це метод, який може одночасно розділяти декілька джерел звуку на окремі аудіопотоки без нагляду[4] . Однак, вибір правильного аудіопотоку з безлічі вихідних потоків залишається викликом і вимагає знання голосової команди користувача зверху вниз. На додаток до використання тригерної фрази “Hey Siri” як сильного акустичного сигналу для ідентифікації цільового потоку, розроблений конкурентний підхід для розділення дикторів і систему вибору глибокого потоку навчання.

Розгорнутий легкий, обчислювальний метод навчання без нагляду для сліпого розділення джерел, щоб розкласти масив сигналів мікрофонів на незалежні аудіопотоки. Метод використовує статистичну властивість незалежності між конкуруючими джерелами і співвідношенням спектральних компонентів кожного джерела, є чисельно стабільним і швидкозбіжним. Алгоритм BSS вивчає акустичний канал на кожному новому вхідному звуковому кадрі безпосередньо з мікрофонних сигналів і відокремлює конкуруючі джерела від обчислювальної складності $O(M^2)$ на джерело в даному піддіапазоні.

Він виводить N_s потоки M вхідних каналів. Хоча алгоритм не страждає від неоднозначності перестановок частоти, відомої в інших традиційних підходах, таких як незалежний компонентний аналіз, розташування вихідних джерел не може бути відомо заздалегідь.

Алгоритми придушення шумових і залишкових ехо-сигналів та розділення джерел, відтворюють додаткову роль у запропонованій системі. Вони спільно вивчають акустичне середовище і визначають різноманітні умови, в яких може використовуватися HomePod, наприклад, тиху обстановку, гучний фон, гучне відтворення та конкуруючий голос. Оскільки попередню інформація про акустичну сцену важко отримати під час виконання, а також в розташуванні Ns. Щоб визначити найкращий аудіопотік за допомогою голосового тригера розробилена система вибору потоків на основі глубокого навчання без нагляду. Система використовує детектор DNN "Hey Siri", постійно відстежує Ns+1 аудіопотоків. Коли виявляється "Hey Siri", кожному потоку присвоюється оцінка. Вибирається потік з найвищим показником і надсилається до Siri для розпізнавання мови та завершення завдання.

На (рис. 1.2) зображені приклади спускових моментів, підрахованих під час “Hey Siri” для зменшення шуму на основі маски у синьому кольорі та найкращого потоку BSS в зеленому. Лівий графік показує дані, отримані у відносно тихому акустичному стані. Добре видно, що зменшення шуму на основі маски є оптимальним. Дані, показані на правому графіку, отримані у складному акустичному середовищі – мовний сигнал, пошкоджений гучним телевізором, що грає поруч. Потік BSS має високу оцінку і повинен бути обраний для зменшення шуму на основі маски.

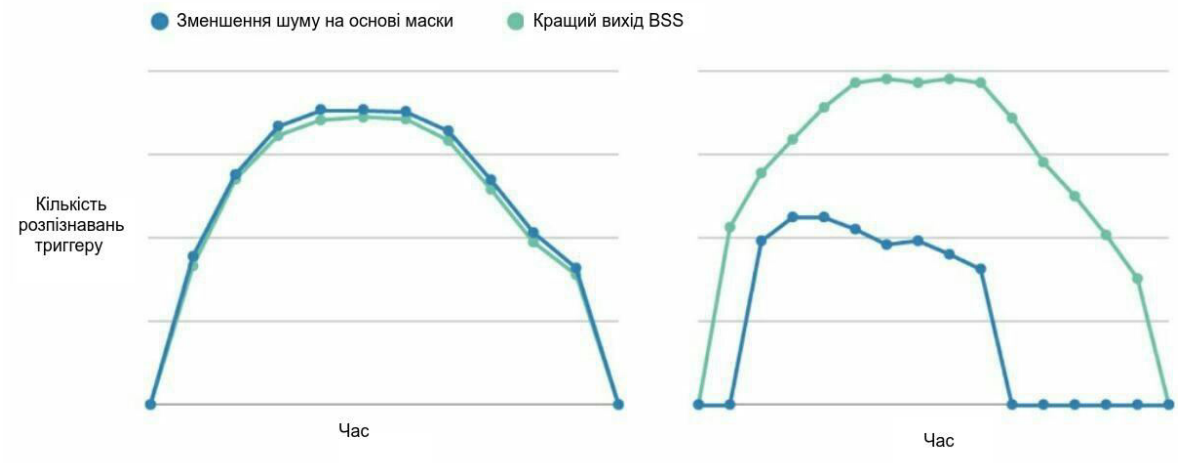


Рисунок 1.2 – Оцінка голосового сигналу у порівнянні з часом: голосова команда Siri, записана під час відтворення музики низького рівня на HomePod(ліворуч), голосова команда Siri, записана на HomePod з програванням телевізора у фоновому режимі(праворуч).

Голосова команда Siri, записана в присутності гучної музики:

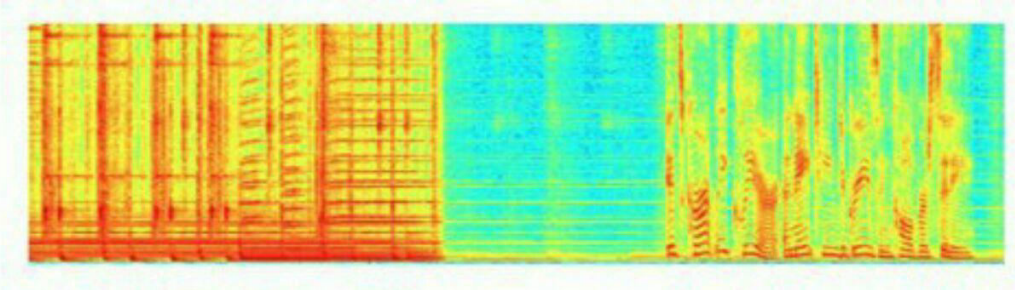


Рисунок 1.3 – Сигнал мікрофона

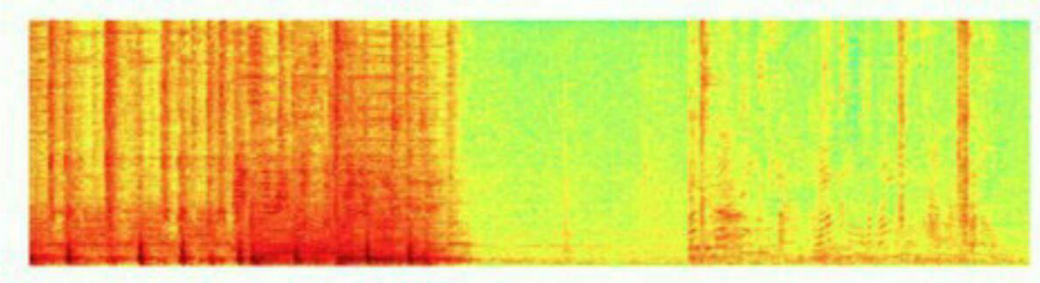


Рисунок 1.4 – вихід MCEC

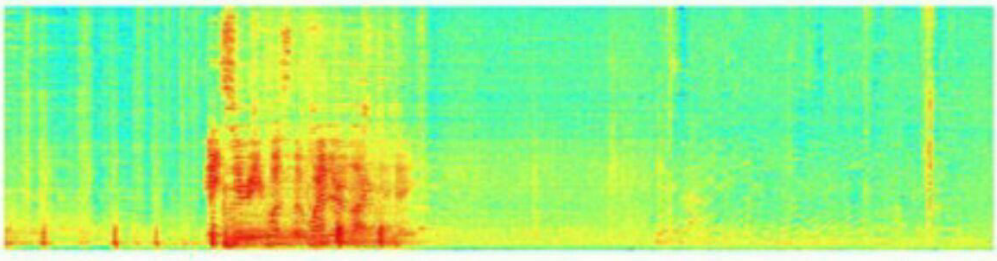


Рисунок 1.5 – Сигнал, посилений придушенням ехо-маски

1.3. Формалізація постановки задачі для дослідження

Задачею першого розділу бакалаврського проекту є дослідження актуальності проблеми, що стосується аналізу та обробки вхідної сенсорної інформації у форматі цифрового аудіо сигналу, вивчення новітніх технологій,

готових рішень та програмних модулів, що якимось чином торкаються до вирішення данної проблеми.

Також метою даного дослідження є вивчення теоретичного підґрунтя описаних систем та проведення аналізу впливів різноманітних факторів.

Система може бути повністю інтегрованою в апаратний пристрій або працювати як незалежне програмне забезпечення.

2. ОПИС АЛГОРИТМІВ ФІЛЬТРУВАННЯ АКУСТИЧНОГО СИГНАЛУ

У запропонованому підході зосереджуємося на додатках реального часу (наприклад, відеоконференціях) з низькою складністю. Також, зосереджуємося на повному діапазоні (48 кГц) мови. Для досягнення цих цілей ми вибираємо гібридний підхід, де ми покладаємося на перевірені методи обробки сигналів і використовуємо глибинне навчання для заміни оцінювачів, які традиційно важко було правильно налаштувати. Цей підхід контрастує з так званими end-to-end системами, де більшість або всі операції обробки сигналів замінюються машинним навчанням. Системи end-to-end чітко продемонстрували можливості глибокого навчання, але результат роботи цих систем є можливим за рахунок значно підвищеної складності.

В данній роботі показується, що запропонований підхід має прийнятну складність і що він забезпечує кращу якість, ніж більш традиційні підходи. В розділі 4 підсумовується з напрямами подальшого вдосконалення цього підходу.

2.1. Моделювання акустичного сигналу

Розглянемо гібридний підхід до придушення шуму. Мета полягає в тому, щоб використовувати глибоке навчання для аспектів придушення шуму, які вимагають ретельного налаштування та при використанні основних блоків обробки сигналів для частин, які цього не потребують. Основний цикл обробки заснований на вікнах 20 мс з 50% перекриттям (зсув 10 мс). Як аналіз, так і синтез використовують одне і те ж вікно Vorbis [10], яке задовольняє критерію Princen-Bradley [11]. Вікно визначається як:

$$w(n) = \sin\left[\frac{\pi}{2} sin^2\left(\frac{\pi n}{N}\right)\right]$$

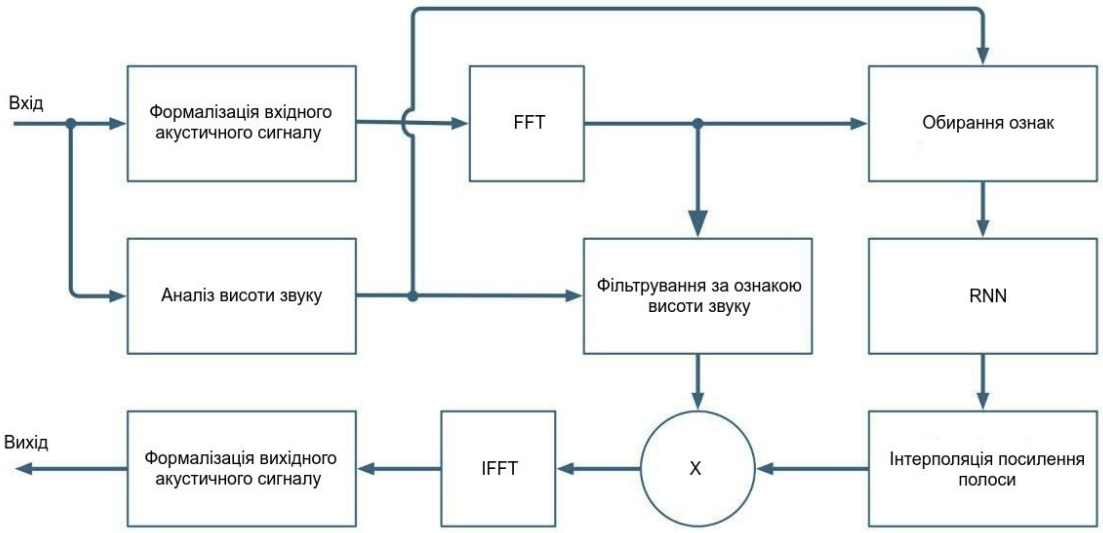


Рисунок 2.1 – Блок-схема рівня сигналу для системи

Основна частина придушення виконується на спектральній оболонці з низькою роздільною здатністю з використанням ваги, обчисленої рекурентною нейронною мережею (RNN). Ці ваги - це квадратний корінь ідеальної маски співвідношення (IRM). Більш точний крок придушення послаблює шум між гармоніками тону, використовуючи фільтр гребінця.

У підході, застосованому в [5], нейронна мережа використовується для безпосередньої оцінки величин частотних контейнерів і вимагає в цілому 6144 прихованих одиниць і близько 10 мільйонів ваг для обробки 8 кГц мови. Масштабування до 48 кГц мовлення з використанням кадрів 20 мс вимагало б мережу з 400 виходами (від 0 до 20 кГц), що явно призводить до більш високої складності, ніж ми можемо собі дозволити.

Одним із способів запобігти цієї проблеми є припущення, що спектральні криві мовлення та шуму є достатньо плоскими, щоб використовувати більш грубу роздільну здатність, ніж частотні контейнери. Крім того, замість безпосередньої оцінки спектральних величин, оцінюються ідеальні коефіцієнти критичних смуг, котрі обмежені діапазоном від нуля до одиниці.

Вирішено розділити спектр за наближенням до шкали Барка [12]. Тобто смуги слідують за шкалою Барка на високих частотах, але завжди мають принаймні 4 контейнера на низьких частотах. Замість прямокутних діапазонів використовуються трикутні смуги, причому пікова відповідь знаходиться на межі між смугами. Тому в мережі потрібно лише 22 вихідних значення в діапазоні [0; 1].

Нехай $w_b(k)$ – амплітуда смуги b на частості k, ми маємо $\sum_b w_b(k) = 1$. Для перетвореного сигналу $X\ (k)$ енергія в смузі задається :

$$E(b) = \sum_k w_b(k)|X(k)|^2$$

Коефіцієнт ваги для кожної групи визначається як g_b

$$g_b = \sqrt{\frac{E_s(b)}{E_x(b)}}$$

де $E_s(b)$ - енергія чистої (істинної) мовлення, а $E_x(b)$ - енергія вхідної (шумної) мовлення. Враховуючи ідеальний коефіцієнт посилення $\widehat{g_b}$, до кожного частоного контейнеру застосовується інтерпольований коефіцієнт:

$$r(k) = \sum_b w_b(k)\widehat{g_b}$$

Основний недолік використання похідних смуг від Барка для обчислення коефіцієнта підсилення є те, що ми не можемо моделювати більш тонкі деталі спектру.

На практиці це запобігає придушенню шуму між гармоніками тону. Як альтернатива, ми можемо використовувати гребінчастий фільтр на періоді висоти звуку, щоб скасувати міжгармонічний шум аналогічно тому, як функціонують пост-фільтри мовного кодека.

Оскільки періодичність мовного сигналу в значній мірі залежить від частоти (особливо для частоти дискретизації 48 кГц), поле фільтрації працює в частотній області на основі коефіцієнта фільтрації по смузі a_b . Нехай $P\ (k)$ -

віконний DFT сигналу $x\ (n\ -\ T)$ із затримкою кроку, фільтрація виконується шляхом обчислення $X\ (k)\ +\ a_b\ P\ (k)$, а потім перенормуванням отриманого сигналу має мати однакову енергію в кожній смузі так як і оригінальний сигнал $X\ (k)$.

Кореляція тону для смуги b визначається як:

$$p_b = \frac{\sum_k w_b(k) R[X(k)P^*(k)]}{\sqrt{\sum_k w_b(k)|X(k)|^2 \cdot \sum_k w_b(k)|P(k)|^2}},$$

де $R[\cdot]$ – позначає дійсну частину комплексного значення, та $\cdot^*$ – є комлексним спряженням.

Отримання оптимальних значень коефіцієнта фільтрації a_b є складним, а значення, що мінімізують середню квадратичну помилку, не є перцептуально оптимальними. Замість цього, використовується підхід, виялений на основі наступних обмежень і спостережень. Оскільки шум викликає зменшення кореляції висоти, ми не очікуємо, що p_b буде в середньому більше g_b, тому для будь-якої смуги, яка має $p_b\ \geq\ g_b$, ми використовуємо $a_b\ =\ 1$. Коли шум відсутній, потрібно уникнути спотворення сигналу, тому при $g_b = 1$ визначаємо $a_b\ =\ 0$. Аналогічно, коли $p_b\ =\ 0$, крок для поліпшення висоти звуку відстуній, тому $a_b\ =\ 0$. Наступний вираз для коефіцієнта фільтрації враховує всі ці обмеження з плавною поведінкою між ними:

$$\alpha_b = \min\left(\sqrt{\frac{p_b^2(1-\ g_b^2)}{(1-\ p_b^2)g_b^2}}, 1\right)$$

Незважаючи на те, використовується фільтр FIR, можливо обчислити $P\ (k)$ на основі фільтра IIR, за формулою $H\ (z)\ =\ 1/(1-\beta_z^{-T})$, що призводить до збільшення загасання між гармоніками за рахунок дещо збільшеного спотворення.

Наступний етап має сенс лише для того, щоб вхід мережі включав логування спектру шумового сигналу на основі тих самих смуг, що використовуються для виведення. Для поліпшення кондиціонування

тренувальних даних застосовується DCT на спектрі логу, в результаті чого утворюються двадцять два коефіцієнти BFCC. На додаток до цього, включається перша похідна за часом і друга похідна за часом перших 6 BFCC. Оскільки потрібно обчислити висоту звуку, обчислюється DCT кореляції тону через смуги частот і включаються до набору ознак перші 6 коефіцієнтів.

На данному етапі ми включаємо період висоти звуку, а також спектральну нестаціонарну метрику, яка може допомогти у виявленні мови. Загалом використовується 42 вхідні функції. На відміну від функцій, що зазвичай використовуються в розпізнаванні мови, ці функції не використовують нормалізацію капстру, але включають перший коефіцієнт кепстру. Вибір є продуманим з огляду на те, що ми повинні відстежувати абсолютний рівень шуму, але він призводить до того, що функії стають чутливими до абсолютної амплітуди сигналу і до частотної характеристики каналу.

2.2. Архітектура глибинного навчання

Нейронна мережа слідує за традиційною структурою алгоритмів подавлення шуму, як показано на (рис. 2.2) Конструкція базується на припущенні, що три рекурентні шари кожен з них відповідає за один з основних компонентів з рис.1. Звичайно, на практиці, нейронна мережа може вільно відхилятися від цього припущення (і, ймовірно, до деякої міри). Вона включає в себе в загальній складності 215 частин, 4 прихованих шари, причому найбільший шар має 96 частин. Виявилено, що збільшення кількості частин суттєво не покращує якість подавлення шумів. Однак функція втрат і спосіб побудови даних навчання мають великий вплив на якість кінцевого результату.

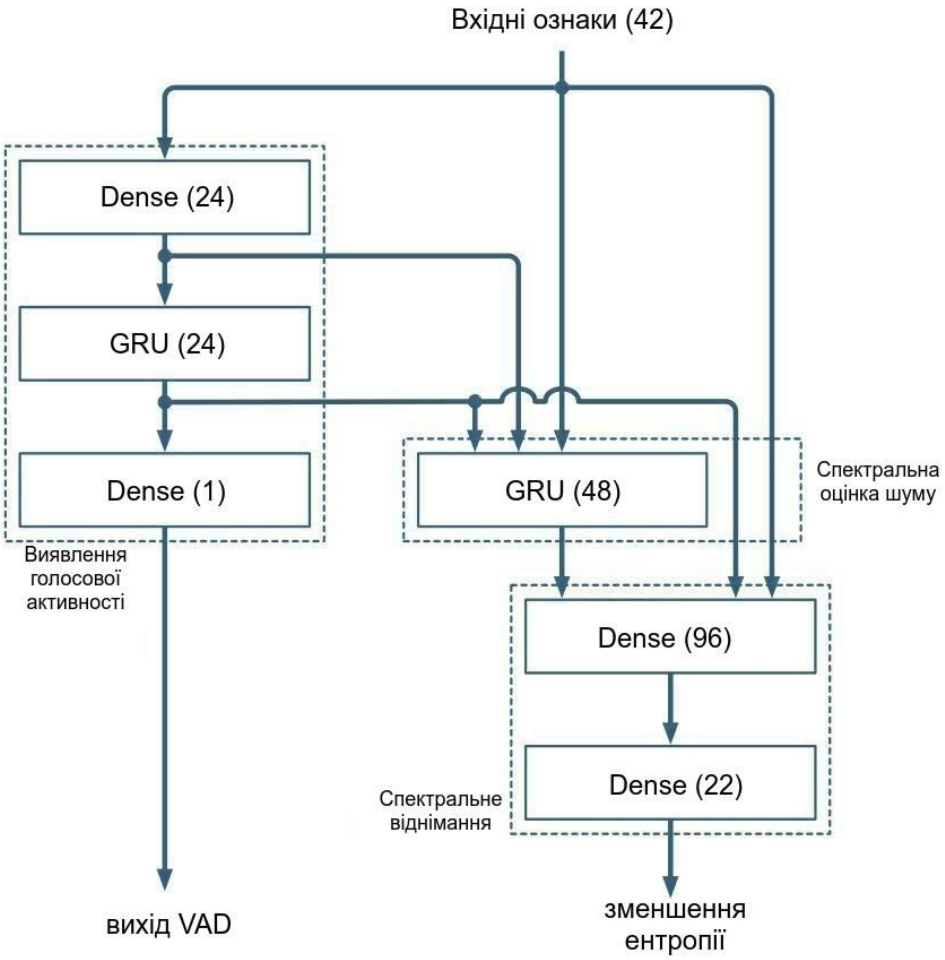


Рисунок 2.2 – Архітектура нейронної мережі

Оскільки потрібні коректні значення коефіцієтів ваги, дані для навчання мають містити як мовлення із шумом, так і зразки чистої мови. Тренувальні дані повинні бути побудовані штучно шляхом додавання шуму до даних зі зразками чистого мовлення. Для мовних даних ми використовуємо базу даних мовлення McGill TSP (французьку та англійську) та багатомовну базу даних мови NTT для телефонометрії (21 мова) Використовуються різні джерела шуму, включаючи комп'ютерні вентилятори, офіс, натовп, літак, автомобіль, поїзд, будівництво. Шум змішується на різних рівнях для отримання широкого діапазону співвідношення сигнал / шум, включаючи чисті сегменти мовлення та шуму.

Оскільки в системі не використовується середня нормалізація кепстру, потрібно збільшити об’єм даних для навчання, щоб зробити мережу надійною

щодо змін частотних відповідей. Це досягається шляхом фільтрації шуму і мовного сигналу незалежно для кожного прикладу навчання, використовуючи фільтр другого порядку за формулою:

$$H(z) = \frac{1 + r_1 z^{-1} + r_2 z^{-2}}{1 + r_3 z^{-1} + r_4 z^{-2}},$$

де кожен з r_1 ... r_4 – випадкові значення, рівномірно розподілені в діапазоні $\left[-\frac{3}{8}, \frac{3}{8}\right]$. Надійність амплітуди сигналу досягається зміною кінцевого рівня змішаного сигналу. В цілому, є 6 годин мовлення і 4 години шумових даних, використовуючи різні комбінації ваг, фільтрів та дискретизуючи дані на частоти від 40 кГц до 54 кГц в результаті утворюється 140 годин мовлення з шумом.

Функція втрат, яка використовується для навчання, визначає як мережа зважує надмірне послаблення в порівнянні з недостатнім послабленням, коли неможливо точно визначити правильні коефіцієнти ваг. Незважаючи на те, що при оптимізації значень в діапазоні $[0, 1]$ використовується функція двійкової крос-ентропії, це не дає хороших результатів для ваг, оскільки не відповідає їх сприйнятливому ефекту. Для оцінки коефіцієнта підсилення $\widehat{g_b}$ та основної істини g_b використовуємо для тренування функцію втрат:

$$L(g_b, \widehat{g_b}) = (g_b^{\gamma} - \hat{g}_b^{\gamma})^2,$$

де показник γ є перцептивним параметром, який керує тим, наскільки агресивно подавлювати шум.

Оскільки $\lim_{\gamma \to 0} \frac{x^{\gamma}-1}{\gamma} = \log(x)$, $\lim_{\gamma \to 0} L(g_b, \widehat{g_b})$ мінімізує середньоквадратичну похибку лог-енергії, що зробить фільтрацію занадто агресивним, та призведе до відсутності ваги. На практиці, величина γ = 1/2 забезпечує хороший компроміс і еквівалентна мінімізації середньої квадратичної похибки енергії, підвищеної до потужності 1/4. Іноді, при вмиканні вхідного сигналу або при високій частоті, коли сигнал фільтрується

низькочастотно може бути виялена відсутність шуму та відсутність мовлення в певному діапазоні. Коли це відбудеться, фактичні дані для цього випробування явно позначаються як невизначені, а функція втрати для цього коефіцієнта ігнорується, щоб уникнути шкоди процесу навчання.

Для виходу VAD мережі використовуться стандартна функцію крос-ентропії. Навчання виконується за допомогою бібліотеки Keras з базовою станцією Tensorflow.

При використанні коефіцієнтів $\hat{g}_b$ для видалення шуму, вихідний сигнал іноді може звучати надто сухим, не маючи мінімального очікуваного рівня реверберації. Проблема легко усувається шляхом обмеження розпаду $\widehat{g_b}$ між кадрами. Зглажені коефіцієнти $\tilde{g}_b$отримані за формулою:

$$\tilde{g}_b = \max\left(\lambda \tilde{g}_b^{(prev)}, \hat{g}_b\right)$$

де $\tilde{g}_b^{(prev)}$ – фільтрований коефіцієнт підсилення попереднього кадру і коефіцієнт розпаду λ = 0.6 еквівалентний часу реверберації 135 мс.

Щоб полегшити розгортання алгоритмів видалення шуму, бажано, щоб розмір і складність системи були низькими. Розмір виконуваного файлу переважає 87,503 ваг, необхідних для представлення 215 частин у нейронних мережах. Щоб зберегти розмір як можна менше, ваги можуть бути квантовані до 8 біт без втрати продуктивності. Це дозволяє підібрати всі ваги в кеш-пам'яті L2 процесора.

Оскільки кожна вага використовується точно один раз на кадр в операції множення-додавання, нейронна мережа вимагає 175000 операцій з плаваючою комою (ми рахуємо множення як дві операції) на кадр, так 17,5 Mflops для використання в реальному часі. IFFT та два FFT за кадр вимагають близько 7,5 Mflops, пошук висоти звуку (який працює на 12 кГц) вимагає близько 10 Mflops. Загальна складність алгоритму становить близько 40 мфлоп, що порівнянно з поширеним кодером мовлення.

Невекторизована C реалізація алгоритму вимагає близько 1,3% єдиного ядра x86 (Haswell i7-4800MQ) для виконання придушення шуму 48 кГц з одного каналу. Складність одного і того ж коду з плаваючою точкою в реальному часі на ядрі ARM Cortex-A53 становить 1,2%.

3. СИСТЕМА ВИДАЛЕННЯ ШУМУ З АУДІОФАЙЛУ

У цьому розділі описується процес розробки, фази формалізації архітектури, кодування та тренування нейронної мережі результатом яких є система фільтрування акустичного сигналу.

Проблема розпізнавання акустичного сигналу була та залишається однією з найважчих завдань у машинному навчанні. Традиційні підходи передбачають ретельний аналіз та вилучення звукових ознак, які відокремлюють одну фонему від іншої. Для того, щоб реалізувати подібні алгоритми, потрібно мати глибокий досвід в роботі з даними та обробкою сигналів. Складність навчального процесу змусила групи дослідників шукати альтернативні, більш автоматизовані підходи.

Зі зростанням розвитку глибинного навчання, потреба в ручних функціях зменшилася. Навчальний процес нейронної мережі зазнав значного спрощення. У наш час, є можливим подавати на вхід акустичні сигнали або у їх необробленій формі, або у вигляді спектрограм та дивитися на покращення моделі.

Метою є створення системи веб-служби, яка надає доступ до API. Дозволяти приймати акустичні сигнали, закодовані як масиви чисел з рухомою комою. В результаті роботи системи очікується отримати відфільтрований аудіосигнал.

План етапів розробки:

- Отримати набір даних для навчання моделі
- Розробка архітектури моделі
- Реалізація разом з модульними тестами
- Тренування на визначеному наборі даних
- Тестування та виміри точності

Для розробки було обрано скористатися перевагами проекту з відкритим кодом “**Common Voice”**, котрий зосереджується на зборі великих наборів даних, які можуть використовуватися у різних проектах розпізнавання мовлення.

Набори даних складаються з файлів у форматі .**waw** та транскрипцій тексту. Поняття часового вирівнювання відсутнє. Присутнім є лише акустичний сигнал і текст для кожного висловлювання.

Складність нейронною мережі, процес її навчання прямо пропорційні складності поставленої задачі. Вона повинна бути здатна виражати більш складні закономірності в даних. Чим потужнішою є мережа, тим простіше вона запам'ятатовує сценарії навчання. Данна ситуаціє є небажаною і призводить до перенасичення нейронної мережі. Щоб зменшити ймовірність перенасичення, потрібно збільшити масиви даних та зібрати більше «реальних» прикладів. Додаткові набори даних, які ми будемо використовувати, добре відомі спільнотою розробників:

- **LibriSpeech**
- **VoxForge**

Ці два набори даних є одними з найбільш популярних, які знаходяться у вільному доступі. Звичайно в мережі присутні також інші набори, які було вирішено опустити, оскільки вони займають досить багато пам’яті на диску. Після завантаження і попередньої обробки двох інформаційних наборів, описаних вище, можливо перейти до налаштування нейронної мережі.

Незважаючи на те, що чимало традиційних задач буде здійснюватися нейронною мережею автоматично, все ще необхідно чітко контролювати процес та розуміти що відбувається заради обґрунтування її гіперпараметрів.

Окрім того, вирішено обробляти акустичні сигнали у формі нейронної мережі з мінімальною можливою складністю. Це дозволить знизити вимоги до пам'яті та значно скоротити час, необхідний для зближення параметрів моделі

до теоритично ідеальних, використання яких призводить до очікуваних результатів роботи системи.

Розглянемо представлення акустичного сигналу, завантаженого з файлу wawe в програмі, з .**waw** файлу:

```
01. import librosa
02. import librosa.display
03.
04. SAMPLING_RATE=16000
05.
06. # ...
07.
08. wave, _ = librosa.load(path_to_file, sr=SAMPLING_RATE)
09.
10. librosa.display.waveplot(wave, sr=SAMPLING_RATE)
```

Наведений вище код вказує, що ми хочемо завантажити аудіодані з частотою дискретизації 16k. Завантаживши дані, відображаємо їх по осі за часом:

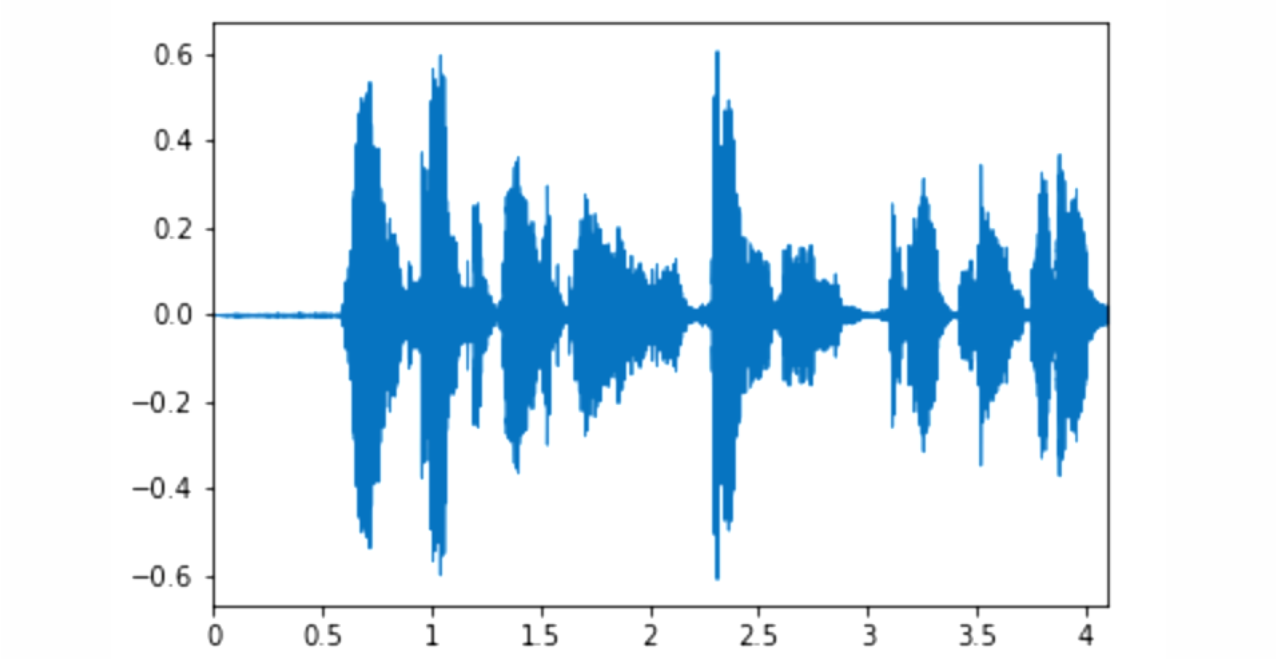


Рисунок 3.1 – Цифрове відображення акустичного сигналу

Вісь X явно показує час, Y – амплітуду. З рис. 3.1 видно, що ми маємо негативні значення в сигналі. Амплітуда являє собою максимальну різницю переміщень фізичного об'єкта, під час воливань. У нашому випадку, аудіосигнал - це не що інше, як коливання повітря. Під час коливання повітря, очевидно потрібна **точка відліку**. Це, у свою чергу, дозволяє визначити точну специфіку коливання - як вона «піднімається» над точкою відліку, а потім

повертається назад. Уявімо, що електрична схема дає вихідний сигнал в діапазоні від -1V до 1V . Щоб завантажити його в цифровому вигляді, потрібно захопити ці значення в дискретні моменти часу. Частота дискретизації – це не що інше, як кількість разів в секунду, коли буде виміряно та збережене значення шумометра.

Створимо машину генерації сигналів, яка виведе синусоїдальну задану частоту і амплітуду:

```
01. def gen_sin(freq, amplitude, sr=1000):
02.     return np.sin(
03.         (freq * 2 * np.pi * np.linspace(0, sr, sr)) / sr
04.     ) * amplitude
```

Так виглядє одна тисяча точок сигналу для частоти 30 і амплітуди 1:

```
01. import seaborn as sns
02.
03. sns.lineplot(data=gen_sin(30, 1))
```

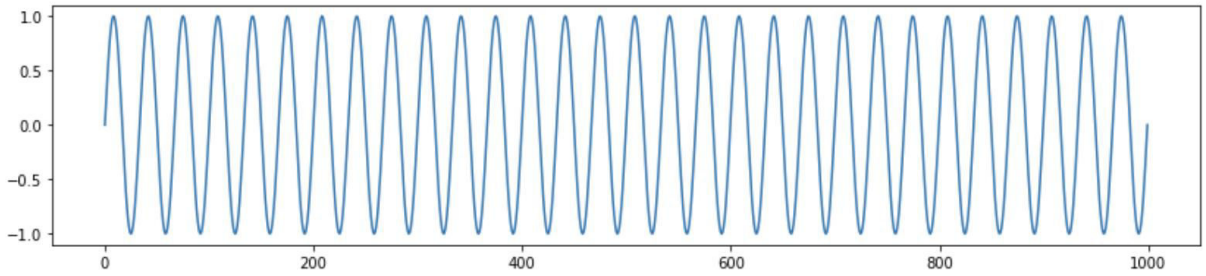


Рисунок 3.2 – Графік синусоїди з частотою 30 та амлітутою 1

Так виглядає синусоїда сигналу для частоти 10 і амплітуди 0.6:

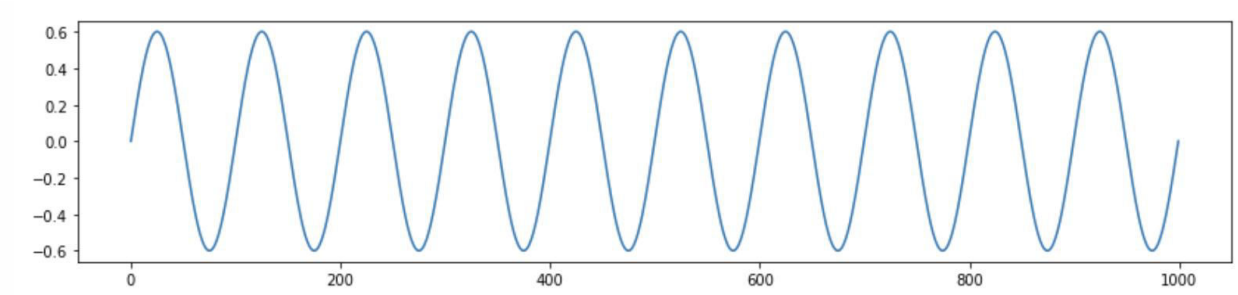


Рисунок 3.3 – Графік синусоїди з частотою 10 та амлітутою 0.6

З рис. 3.2, 3.3 можливо підрахувати кількість разів, коли значення ділянок наближаються до свого максимуму. Знаючи, що синус має тільки один максимум протягом свого періоду і те, що ми показуємо лише одну секунду.

Розглянемо випадок, що було б отримано в результаті, якби були підпраховані синусоїдальні сигнали з різною частотою та амплітудою. На рис. 3.4 можливо побачити 3 різні синусоїдальні хвилі, нанесені один на одного, четверта - показує сигнал, який є сумою трьох попередніх:

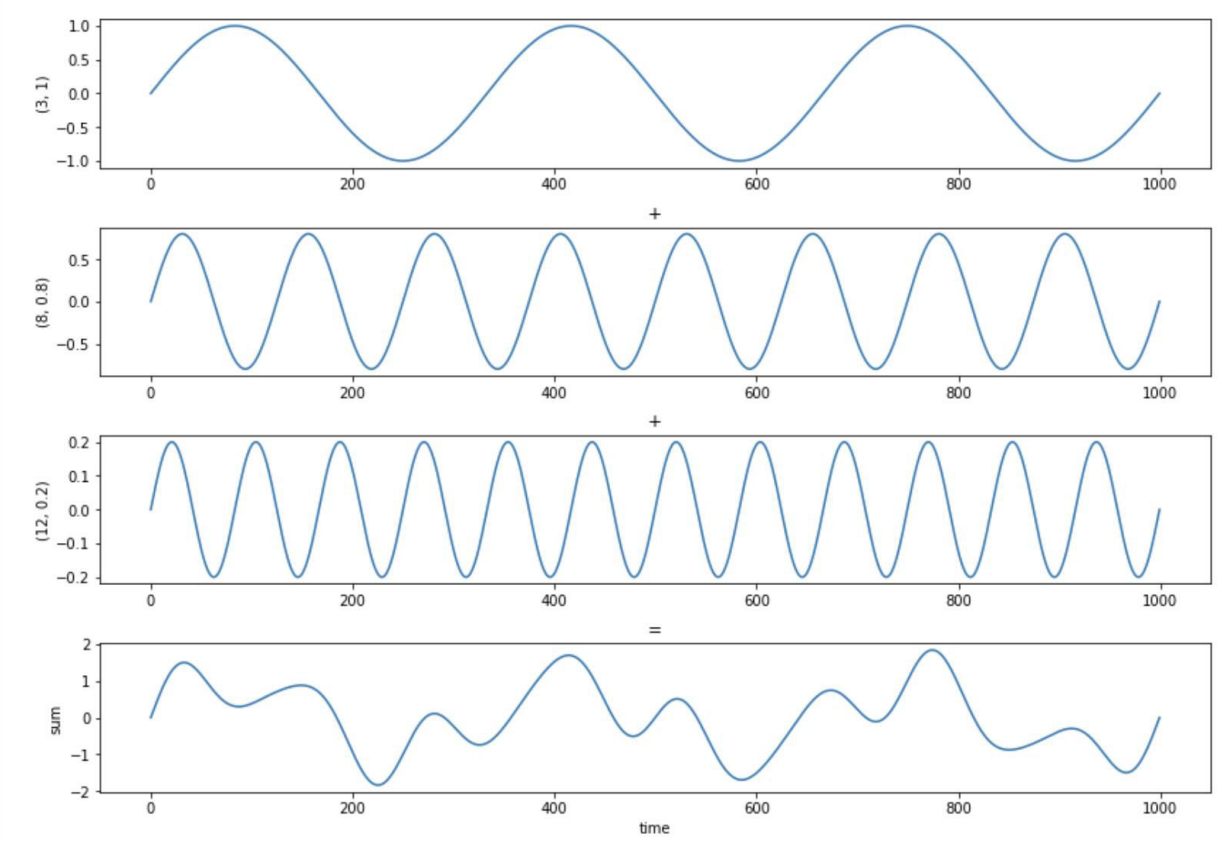


Рисунок 3.4 – Графіки трьох синусоїд з різними частотами, амплітудою та графік їх суми

Ось ще один приклад: останній графік показує суму 5 різних хвиль:

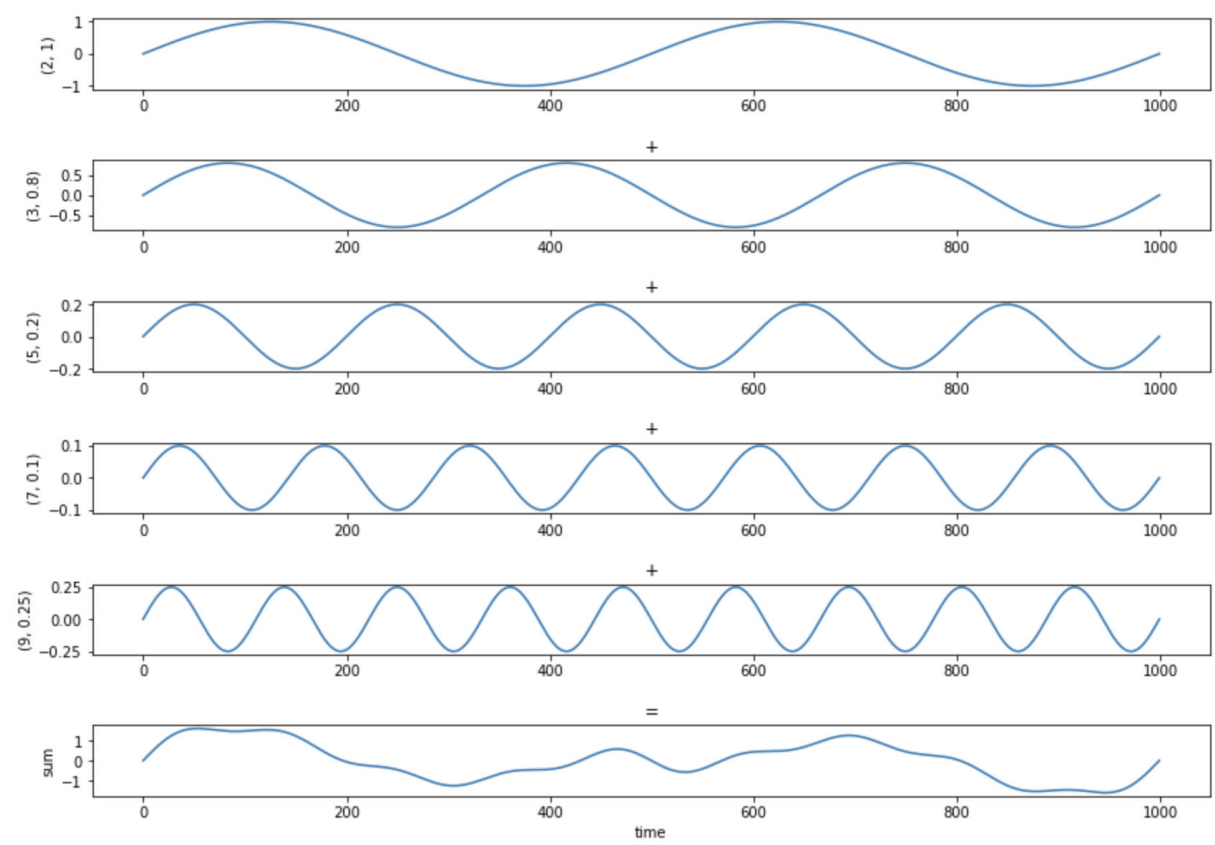


Рисунок 3.5 – Графіки чотирьох синусоїд з різними частотами, амплітудою та графік їх суми

Хвильова комопозиція вже не виглядає систематичною. Виявляється, що є можливим побудувати будь-який сигнал, підсумувавши певну кількість синусоїдних хвиль різних частот і амплітуд . Зворотнє твердження також вірне: будь-який сигнал може бути представлений як сума деякої кількості синусоїдних хвиль різних частот та амплітуд. Це надзвичайно важливо для завдання розпізнавання мовлення та подальшої обробки цифрового сигналу. Частоти відображають це реальну різницю між звуками, що складають фонеми і слова, які ми хочемо розпізнати.

Таким чином з’являється необхідність використовувати перетворення Фур’є . Розглядаються точки даних, які представляють інтенсивність в кожній точці часу та утворюються нові точки даних, що представляють собою

інтенсивність для кожного частотного контейнеру . Загалом, відбувається перетворення області сигналу з часу на частоту . Нехай, фізичний звуковий сигнал будується з частот від 0 до 8000 Гц. Алгоритм FFT розділяє цей повний спектр на контейнери . Діапазон розбивається на 10 контейнерів, таким чином отримуємо проміжки: 0Гц – 800Гц, 800Гц — 1600Гц, 1600Гц – 2400Гц, 2400Гц – 3200Гц, 3200Гц – 4000Гц, 4000Гц – 4800Гц, 4800Гц – 5600Гц, 5600Гц– 6400 Гц, 6400 Гц – 7200 Гц, 7200 Гц – 8 000 Гц.

Розглянемо, як FFT працює на прикладі поданого вище сигналу. Хвилі та ділянки були створені наступною функцією Python:

```
ddef plot_wave_composition(defs, hspace=1.0):
    fig_size = plt.rcParams["figure.figsize"]

    plt.rcParams["figure.figsize"] = [14.0, 10.0]

    waves = [
        gen_sin(freq, amp)
        for freq, amp in defs
    ]

    fig, axs = plt.subplots(nrows=len(defs) + 1)

    for ix, wave in enumerate(waves):
        sb.lineplot(data=wave, ax=axs[ix])
        axs[ix].set_ylabel('{}'.format(defs[ix]))

        if ix != 0:
            axs[ix].set_title('+')

    plt.subplots_adjust(hspace = hspace)

    sb.lineplot(data=sum(waves), ax=axs[len(defs)])
    axs[len(defs)].set_ylabel('sum')
    axs[len(defs)].set_xlabel('time')
    axs[len(defs)].set_title('=')

    plt.rcParams["figure.figsize"] = fig_size

    return waves, sum(waves)
```

Будуємо сигнали та одночасно їх захоплюємо для подальшої обробки:

```
wave_defs = [
        (2, 1),
        (3, 0.8),
        (5, 0.2),
        (7, 0.1),
        (9, 0.25)
    ]
```

Далі обчислимо значення FFT разом з частотами:

```
ffts = np.fft.fft(the_sum)
freqs = np.fft.fftfreq(len(the_sum))

frequencies, coeffs = zip(
    *list(
        filter(
            lambda row: row[1] > 10,
            [ (int(abs(freq * 1000)), coef) for freq, coef in zip(freqs[0:(len(ffts) // 2)], np.abs(ffts)[0:(len(ffts) // 2)]) ]
        )
    )
)

sns.barplot(x=list(frequencies), y=coeffs)
```

Останній виклик створює наступну графік:

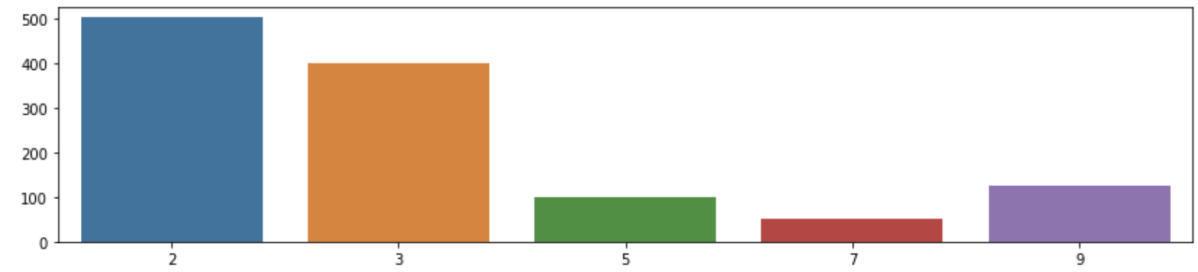


Рисунок 3.6 – Результуючик графік виконання функції plot_wawe_composition

Вісь X представляє тепер частоту в Гц, а вісь Y - інтенсивність. З рис. 3.6 добре видно, що FFT дає нам частоти для всього сигналу, припускаючи, що він періодичний і прямує в часі до нескінченності. Очевидно, коли диктор промовляє «привіт», повітря спочатку вібрує по-різному, змінюється на початку і є ще більш різним в кінці. Потрібно розділити цей звук на невеликі «контейнери» точок даних. Подаючи їх у FFT, ми можемо отримати частоти для кожного з них. Це перетворює область даних з часом на область з частотою в межах контейнеру. Залишається інформація про час на глобальному рівні, що робить наші дані репрезентативними до схеми:

Час – Частота – Інтенсивність

Людське сприйняття є досить комлексним явищем. Враховуючи цей факт, можливо пройти довгий шлях, працюючи над архітектурою моделі розпізнавання, яка наслідує роботу мозку людини, в режимі діалогу дикторів.

Людина відчуває більшу різницю звукових сигналів в межах більш низьких частот(800Гц, 900Гц, 1000Гц). Зі збільшенням частоти різниця буде розпізнаватися все гірше(6100Гц, 6200Гц).

Через це, в 1937 році була створена так звана шкала **Mel**.

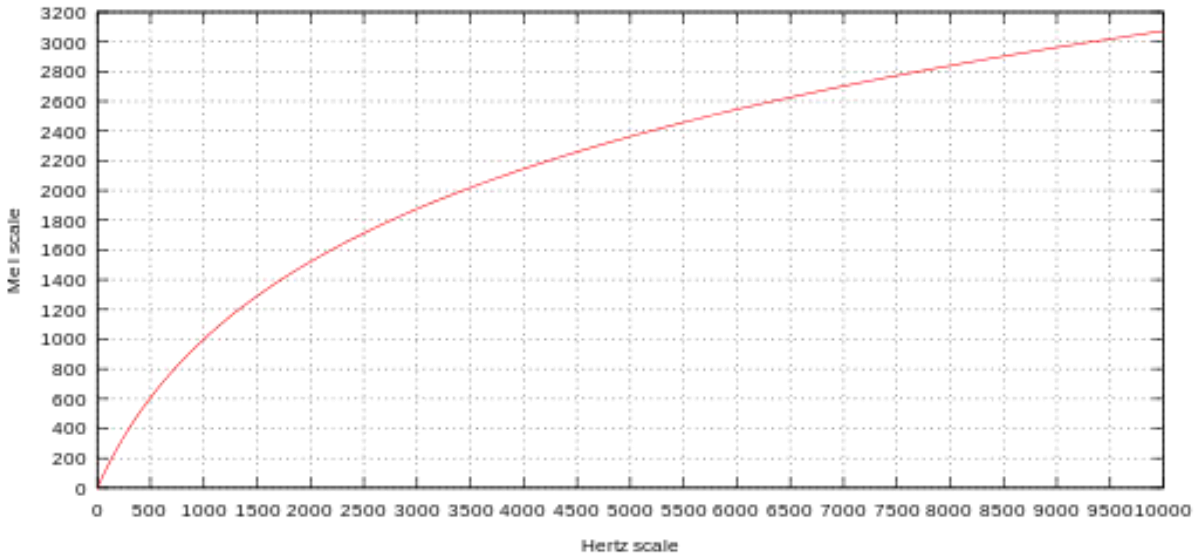


Рисунок 3.7 – Графік залежності висоти звуку в мелах від частоти коливань (для чистого тону)

Багато моделей, які стосуються людської мови, також зменшують важливість інтенсивності, приймаючи журнал повторно масштабованих даних. Результуюча схема залежності часу, частоти та інтенсивності називається **log-Mel** спектрограмою.

Тепер зосередимося на концепціях глибокого навчання, які ми будемо використовувати для побудови і навчання моделі. Традиційно, обробка послідовностей різної складності в глибинному навчанні вирішується рекуррентними нейронними мережами.

Незалежно від вибору різних класифікацій, основна схема завжди однакова: обчислення виконуються послідовно, проходячи через шаблони різних видів протягом певного часу. Оскільки блоки будуть оброблятися один за одним, внутрішній стан повторюваної мережі буде «запам'ятовувати»

специфіку попереднього блоку, включаючи їх у свої майбутні рішення. Форма вихідної інформаціїї буде мати вигляд**: час – частота – рекурентний блок.**

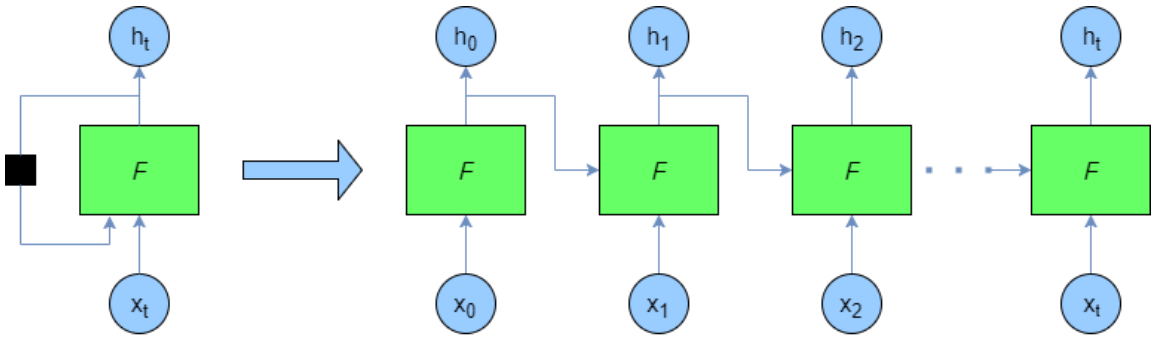


Рисунок 3.8 – Схема конвеєру

Той факт, що обчислення виконуються послідовно, робить їх досить повільними. Пізніші обчислення на конвеєрі (рис 3.8), через їх пряму залежність, більшу частину часу проводять очікуючи на завершення попередніх. Проблема з використанням графічних процесорів ще глобальніша. Їх доцільно використовувати через здатність паралельно виконувати математичні обчислювання на величезних об’ємах даних. З повторюваними мережами ця перевага досить сильно нівелюється.

Передумовою використання RNN є те, що в теорії, вони можуть мати здатність зберігати дуже довгі контексти у своїй «пам'яті». Це твердження нещодавно було випробувано на практиці [9] .

На персональному комп’ютері вкомпонована Nvidia GT 730 з 2 Гб пам'яті для тренування моделей. Проводити місяці з метою очікування збігу повторюванної мережі – не кращий варіант, тому у цьому проекті, вибір пав на дуже ефективну альтернативу – згорткова нейронна мережа .

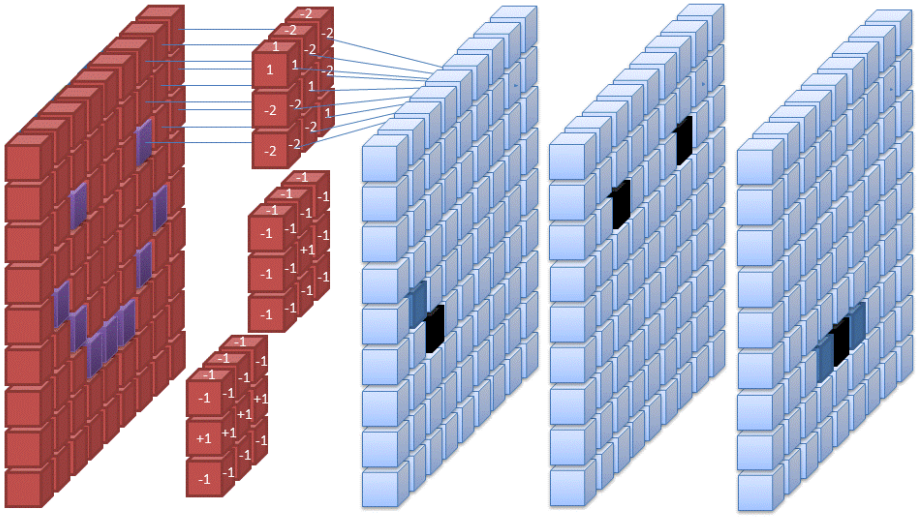


Рисунок 3.9 – Фільтрація в згортковій нейронній мережі

Прості згорткові шари, значною мірою, не використовувалися для обробки послідовностей. Суть обробки послідовностей полягає в тому, щоб мати можливість враховувати більший об’єм контексту. Залежно від роботи, можливо обмежити контекст лише минулим, вивчаючи причинно-наслідкові зв'язки в даних. Рішення для розпізнавання на даний момент полягає у використанні двонаправлених повторюваних шарів. Перший прохід вивчає відносини зліва направо, а другий навчається справа наліво. Результати потім об’єднуються.

Застосовуючи певне заповнення, можливим є легко включити один або двосторонній контекст в 1D витках. Проблема полягає в тому, що для того щоб аналізувати великий контекст – розмір фільтрів потрібно пропорційно збільшувати. Це, у свою чергу, вимагає збільшення об’ємів необхідної пам'яті. Оскільки метою є створення моделі, з невелики затратами на навчання та збереженням невисокої складності системи (з урахуванням зазвичай використовуваних графічних процесорів) GT 730, вимоги до пам’яті мають буди настільки низькими, наскільки це є можливим.

Останнім часом, завдяки успіху **WaveNet**, особливий клас згорткових шарів набув вискої популярності популярності серед спільноти розробників. Варіація називається **Dilated Convolutions** або іноді **Atrous Convolutions**. Розглянемо, як результати залежать від вхідного контексту для простих згорткових шарів:

Нехай, спочатку є лише верхній ряд цифр. Ми збираємося використовувати 1D зв'язки і зробити обґрунтування якомога простішим, кількість фільтрів дорівнює одиниці. Також, для простоти всім значенням фільтрів встановлюються одиниця. На рисунку очевидна перехресна кореляція, оператор приймає 3 значення в контексті, множивши на фільтр і підсумовуючи значення резудьтату: $2 * 1 + 3 * 1 + 4 * 1 = 9$.

1	2	3	4	5	6	7	8	9	10	11	
1	3	6	9	12	15	18	21	24	27	30	
1	4	10	18	27	36	45	54	63	72	81	

Рисунок 3.10 – Приклад згорткового шару

Atrous Convolutions дійсно однакові, за винятком того, що вони розширюють фокус, не збільшуючи розмір фільтра шляхом введення проміжків. Це показано нижче зі згорткою розміру 2 і розширенням 2:

1	2	3	4	5	6	7	8	9	10	11	12
1	2	4	6	8	10	12	14	16	18	20	22
1	2	5	8	12	16	20	24	28	32	36	40

Рисунок 3.11 – Приклад згорткового шару для розміру 2 і розширення 2

Ще один приклад для розміру 2 і розширення 3:

1	2	3	4	5	6	7	8	9	10	11
1	2	3	5	7	9	11	13	15	17	19
1	2	3	6	9	12	16	20	24	28	32

Рисунок 3.11 – Приклад згорткового шару для розміру 2 і розширення 3

Традиційно, згорткові шари супроводжуються активацією *elu (ReLu, Elu, PRelu, Selu). Вони добре вписуються в парадигму «картини відповідності» консенсусів. Рекуренті підрозділи використовують підхід «пам'ятати / забувати». Дві з найчастіше використовуваних реалізацій, GRU і LSTM.

В розробленій моделі з розширеною згорткою, імітується їх здатність «забувати» частини контексту. Для цьоговикористовується підхід «закритих активацій», пояснений у роботі[10]

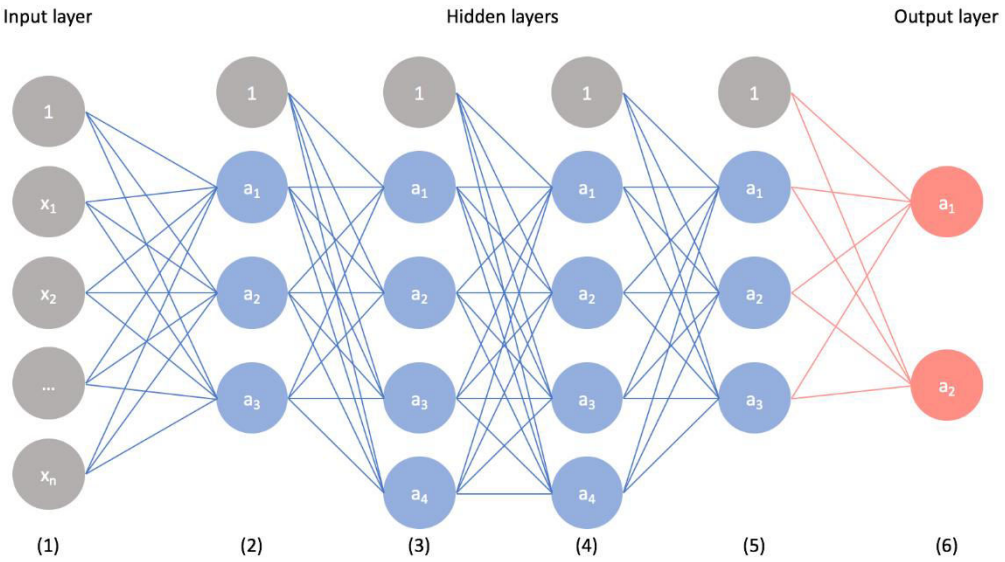


Рисунок 3.12 - Архітектура моделі LSTM мережі

Ідея досить проста: передаємо вхід через модуль Conv1D окремо, застосовуємо модуль tanh і модуль sigmoid відповідно. Результатом є поетапно розібрана аудіо інформація. Для покращення роботи даної моделі, застосуємо модуль tanh ще раз в кінці.

Архітектура системи в цьому проекті буде значною мірою залежати від великого успіху мереж залишкового стилю, та розширених зв'язків. Розробка моделей глибокого навчання не дуже відрізняється від інших типів кодування. Звичайно, процес вимагає спеціальних фонових знань у сфері, але хороші практики кодування залишаються тими ж самими.

Підготовка роботи системи розпізнавання мовлення у текстовому вигляді може займати декільна днів, якщо не тижнів. Уявимо, що у коді є невелика помилка, що перешкоджає процесу знайти хороший локальний мінімум. Це надзвичайно розчаровує, тому що призводить до витрати багато часу на моделі, які тренувалися, але не покращилися.

Почнемо з додавання деяких модульних тестів. У цьому проекті використовується персональний комп'ютер, оскільки поки не маємо наміру

упаковувати розроблену систему в будь-якому місці. Розробка коду ведеться в основному в освітніх цілях. Місце, яке є напрочуд чутливим до помилок, це конвеєр даних. Наприклад, завжди є можливість ввести вхідні вектори, включаючи значення NaN або inf , які через кілька кроків спродукують значення втрат NaN або inf . Додамо простий тест, щоб перевірити першу умову:

```
01. import unittest
02.
03. RUN_TESTS = TRUE
04.
05. class TestNotebook(unittest.TestCase):
06.     def test_it_works(self):
07.         self.assertEqual(2 + 2, 4)
08.
09. if __name__ == '__main__' and RUN_TESTS:
10.     import doctest
11.
12.     doctest.testmod()
13.     unittest.main(
14.         argv=['first-arg-is-ignored'],
15.         failfast=True,
16.         exit=False
17.     )
```

Утворений конвеєр даних, випадково перетасовує кадр даних **Pandas** один раз. Він також створює вибірку методів, які працюють у фоновому режимі, з метою максимальної паралелізації завантаження даних. Виконується завантаження та збільшення даних на ЦП. Також використовується бібліотеку hickle для кешування аудіосигналів на диску. Завантаження файлу хвиль із заданою частотою дискретизації виконується досить повільно. Експерементальним методом було встановлено, що завантаження отриманого масиву чисел з рухомою комою через hickle виконується швидше в 10 разів. Звичайно, для навчання системи потребується прийнятна швидкість подачі даних у мережі, інакше GPU буде простоювати та час виконання конвеєру значно збільшиться.

Функція **random_noise** використовує звукові шуми, включені в **команди мовлення. Ф**ункціея experiment_params являється помічником, який дозволяє легко конструювати хеш-параметри для експерементального навчання:

```
def dataset_params(batch_size=32,
                   epochs=50000,
                   parallelize=True,
                   max_text_length=None,
                   min_text_length=None,
                   max_wave_length=80000,
                   shuffle=True,
                   random_shift_min=-4000,
                   random_shift_max= 4000,
                   random_stretch_min=0.7,
                   random_stretch_max= 1.3,
                   random_noise=0.75,
                   random_noise_factor_min=0.2,
                   random_noise_factor_max=0.5,
                   augment=False):
    return {
        'parallelize': parallelize,
        'shuffle': shuffle,
        'max_text_length': max_text_length,
        'min_text_length': min_text_length,
        'max_wave_length': max_wave_length,
        'random_shift_min': random_shift_min,
        'random_shift_max': random_shift_max,
        'random_stretch_min': random_stretch_min,
        'random_stretch_max': random_stretch_max,
        'random_noise': random_noise,
        'random_noise_factor_min': random_noise_factor_min,
        'random_noise_factor_max': random_noise_factor_max,
        'epochs': epochs,
        'batch_size': batch_size,
        'augment': augment
    }
```

Мітки кодера і декодера

При роботі з функцією втрат **CTC** потрібен спосіб кодувати кожну літеру як числове значення. І навпаки, нейронна мережа дає вираховує вірогідність для кожної букви, задану її індексом у вихідній матриці. Ідея цього підходу полягає в тому, щоб перемістити кодування і декодування до самого мережевого графу. Для цього потрібні дві функції:

- Encode_labels - перетворює рядок на масив цілих чисел
- Decode_codes - перетворює масив цілих чисел у результуючий рядок.

Для цього тестового модулю добре підходить бібліотека **hypothesis** . Її задача полягає в генерації прикладів введення, котрі намагаються фальсифікувати задані припущення:

```
@given(st.text(alphabet="abcdefghijkl234!@#$%^&*", max_size=10))
def test_encode_and_decode_work(self, text):
    assume(text != '')

    params = { 'alphabet': 'abcdefghijkl234!@#$%^&*' }

    label_ph = tf.placeholder(tf.string, shape=(1), name='text')
    codes_op = encode_labels(label_ph, params)
    decode_op = decode_codes(codes_op, params)

    with tf.Session() as session:
        session.run(tf.global_variables_initializer())
        session.run(tf.tables_initializer(name='init_all_tables'))

        codes, decoded = session.run(
            [codes_op, decode_op],
            {
                label_ph: np.array([text])
            }
        )

        note(codes)
        note(decoded)

        self.assertEqual(text, ''.join(map(lambda s: s.decode('UTF-8'), decoded.values)))
        self.assertEqual(codes.values.dtype, np.int32)
        self.assertEqual(len(codes.values), len(text))
```

Ще один фрагмент, який потрібен, це спосіб перетворити сирі акустичні сигнали на log-Mel спектрограми. Таким чином, на графічних процесорах відбудеться приріст продуктивності, а API буде значно простішим.

Для того, щоб ефективно використовувати функцію втрат CTC і декодер, необхідно передавати їй довжину даних, для ефективного представлення фрагментів акустичних сигналів для кожної партії. Це пов'язано з тим, що не всі акустичних сигнали мають однакову довжину, тому доповнюємо їх нулями, щоб виконати перетворення.

Реалізація:

```
def compute_lengths(data,
                        optimizer='Adam',
                        lr=1e-4,
                        alphabet=" 'abcdefghijklmnopqrstuvwxyz",
                        causal_convolutions=True,
                        stack_dilation_rates=[1, 3, 9, 27, 81],
                        stacks=2,
                        stack_kernel_size=3,
                        stack_filters=32,
                        sampling_rate=16000,
                        n_fft=160*4,
                        frame_step=160,
                        lower_edge_hertz=0,
                        upper_edge_hertz=8000,
                        num_mel_bins=160,
                        clip_gradients=None,
                        codename='regular',
                        **kwargs):
```

```
19.         """
20.         Computes the length of data for CTC
21.         """
22.
23.         return tf.cast(
24.             tf.floor(
25.                 (tf.cast(original_lengths, dtype=tf.float32) - params['n_fft']) /
26.                     params['frame_step']
27.             ) + 1,
28.             tf.int32
29.         )
30.         }
```

4. Тестування розробленої системи фільтування акустичного сигналу

Використовуючи TensorBoard, ми отримуємо зручний інструмент для моніторингу прогресу. Розроблена **model_fn** статистика виводу для відстані редагування навчального набору , а також для набору оцінок. Статистика втрат CTC включена за замовчуванням.

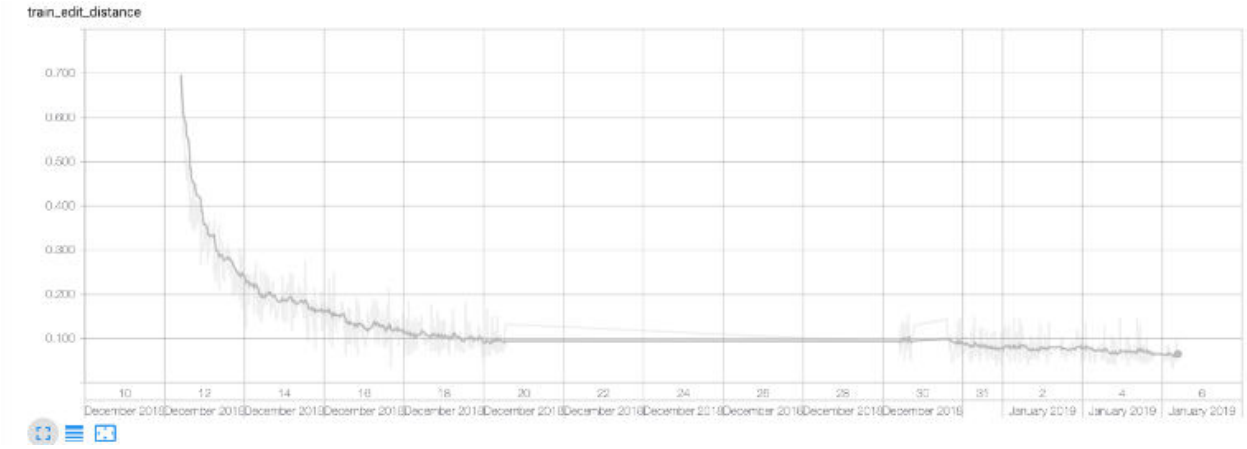


Рисунок 4.1 – Діаграма остаточної моделі

Наступне зображення показує втрату CTC з оранжевою лінією, що представляє оцінку ефективності роботи.

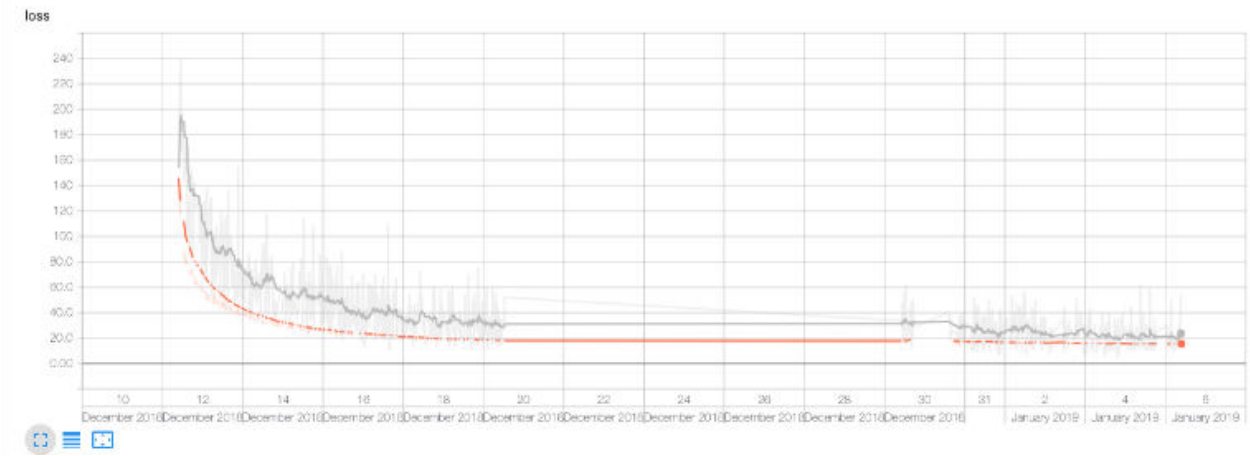


Рисунок 4.2 – Діаграма остаточної моделі з функцією втрат CTC

Для кожної моделі машинного навчання варто оцінити статистичну точність:

```
test(
    dataset_params(
        batch_size=18,
        epochs=10,
        max_wave_length=320000,
        augment=True,
        random_noise=0.75,
        random_noise_factor_min=0.1,
        random_noise_factor_max=0.15,
        random_stretch_min=0.8,
        random_stretch_max=1.2
    ),
    codename='deep_max_20_seconds',
    alphabet=' !"&\',-.01234:;\\abcdefghijklmnopqrstuvwxyz', # !"&',-.01234:;\abcdefghijklmnopqrstuvwxyz
    causal_convolutions=False,
    stack_dilation_rates=[1, 3, 9, 27],
    stacks=6,
    stack_kernel_size=7,
    stack_filters=3*128,
    n_fft=160*8,
    frame_step=160*4,
    num_mel_bins=160,
    optimizer='Momentum',
    lr=0.00001,
    clip_gradients=20.0
)
```

Результат тестування:

```
(...)
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2019-01-07-10:51:09
INFO:tensorflow:Saving dict for global step 1525345: edit_distance = 0.07922124, global_step = 1525345, loss = 13.410753
(...)
```

Це показує, що для тестового набору ми набрали **0.079** відстань для редагування. Щоб назвати приблизну точність інвертуємо значення,та отримуємо **92.1%** - досить не поганий результат.

Гарні результати показав розмір всієї системи: 204MB для моделі, що навчається на наборі даних **375k**+ з агресивним збільшенням (що робить результуючий розмір набору даних в декілька разів більшим).

Тестуємо якість видалення шуму, використовуючи дані мови та шуму, які не використовуються у навчальному наборі. Порівнюємо результати системи з шумопоглиначем на основі MMSE в бібліотеці SpeexDSP. Хоча фільтрація акустичного сигналу працює на частоті 48 кГц, вихідний сигнал має бути відновлений на частоту 16 кГц через обмеження широкосмугового PESQ [16]. Об'єктивні результати на (рис. 4.1), показують значне поліпшення якості за рахунок використання глибокого навчання, особливо для нестаціонарних типів шуму.

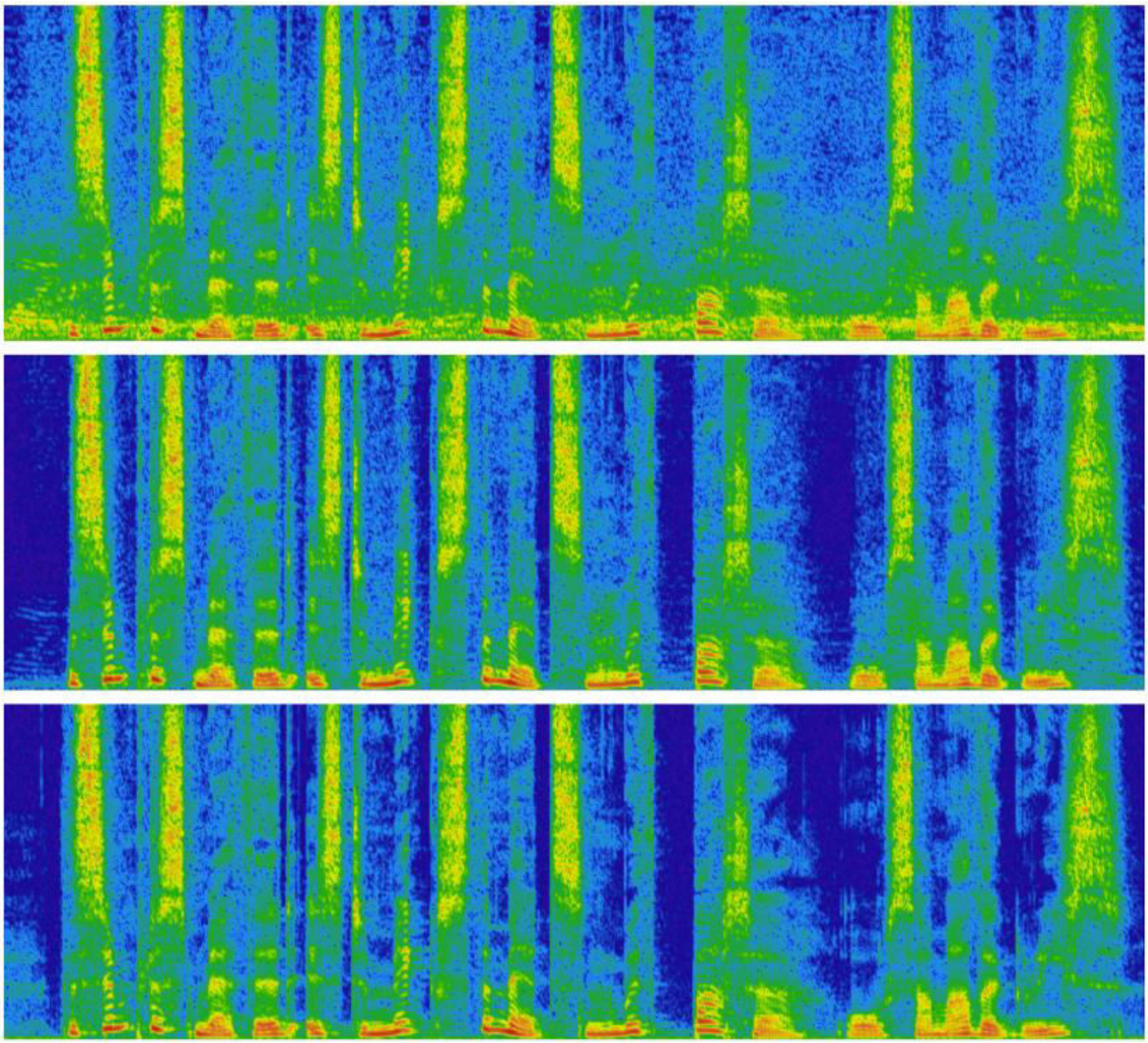


Рисунок 4.3 – Приклад видалення шуму 15дБ SNR. Спектрограма шумного (верхнього), відфільтрованого (середнього) і чистого (нижнього) мовлення. Для ясності показано лише смугу діапазону 0-12 кГц.

ВИСНОВКИ

Метою даного дипломного проекту було створення системи фільтрування акустичного сигналу засобами штучного інтелекту.

В результаті аналізу існуючих рішень було визначено їх основний недолік: закритий програмний код, реалізації схожим систем є вбудованими в апаратний пристрій та являються пропрієтарними.

Аналіз способів ідентифікації аудіофайлів показав доцільність використання гібридного методу цифрової оброки акустичного сигналу з використання нейронних мереж глибинного навчання.

В якості програмних засобів для реалізації системи було використано мову програмування Python та бібліотеки Pandas, hickle та TensorFlow з її інфраструктурними можливостями будувати та тренувати нейронні мережі.

Було вивчено теоретичне підґрунтя декодування аудіофайлів, отримання частотних характеристик сигналу, виділення характерних частот аудіосигналу та фільтрування від нерелеватної аудіо-інформації.

Розроблена система надає можливість виконувати аналіз та обробку аудіофайлів завантажених користувачем і в результаті роботи отримувати відфільтрований акустичний сигнал. Розробка тамкож може виступати в ролі інтерфейсу проміжної обробки акустичних сигналів в комплексних проетках.

У подальшому API системи може бути використане розробниками акустичних систем, апаратних засобів для аудіо-відео конференцій та приладів призначних для запису звуку.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. K. Kinoshita, M. Delcroix, T. Yoshioka, T. Nakatani, A. Sehr, W. Kellermann, and R. Maas. The REVERB challenge: A common evaluation framework for dereverberation and recognition of reverberant speech, *in Proc. IEEE WASPAA*, 2013.
2. The 4th CHiME Speech Separation and Recognition Challenge, 2016. [Електронний ресурс] – Режим доступу до ресурсу: http://spandh.dcs.shef.ac.uk/chime_challenge/chime2016/
3. H. Erdogan, J.R. Hershey, S. Watanabe, M. Mandel, and J. Le Roux. Improved MVDR beamforming using single-channel mask prediction networks, *in Proc. ISCA Interspeech*, 2016.
4. M.S. Pedersen, J. Larsen, U. Kjems, and L.C. Parra. A survey of convolutive blind source separation methods, *Multichannel Speech Processing Handbook*, 2007.
5. A. Maas, Q.V. Le, T.M. O'Neil, O. Vinyals, P. Nguyen, and A.Y. Ng, "Recurrent neural networks for noise reduction in robust ASR," in Proc. INTERSPEECH, 2012.
6. J.R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe. Deep clustering: discriminative embeddings for segmentation and separation, *in Proc. IEEE ICASSP*, 2016.
7. A. Schwartz, C. Hofmann, and W. Kellermann. Spectral feature-based nonlinear residual echo suppression, *in Proc. IEEE WASPAA*, 2013.
8. T. Yoshioka, A. Sehr, M. Delcroix, K. Kinoshita, R. Maas, T. Nakatani, and W. Kellermann. Making machines understand us in reverberant rooms: robustness against reverberation for automatic speech recognition, *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 114–126, 2012.

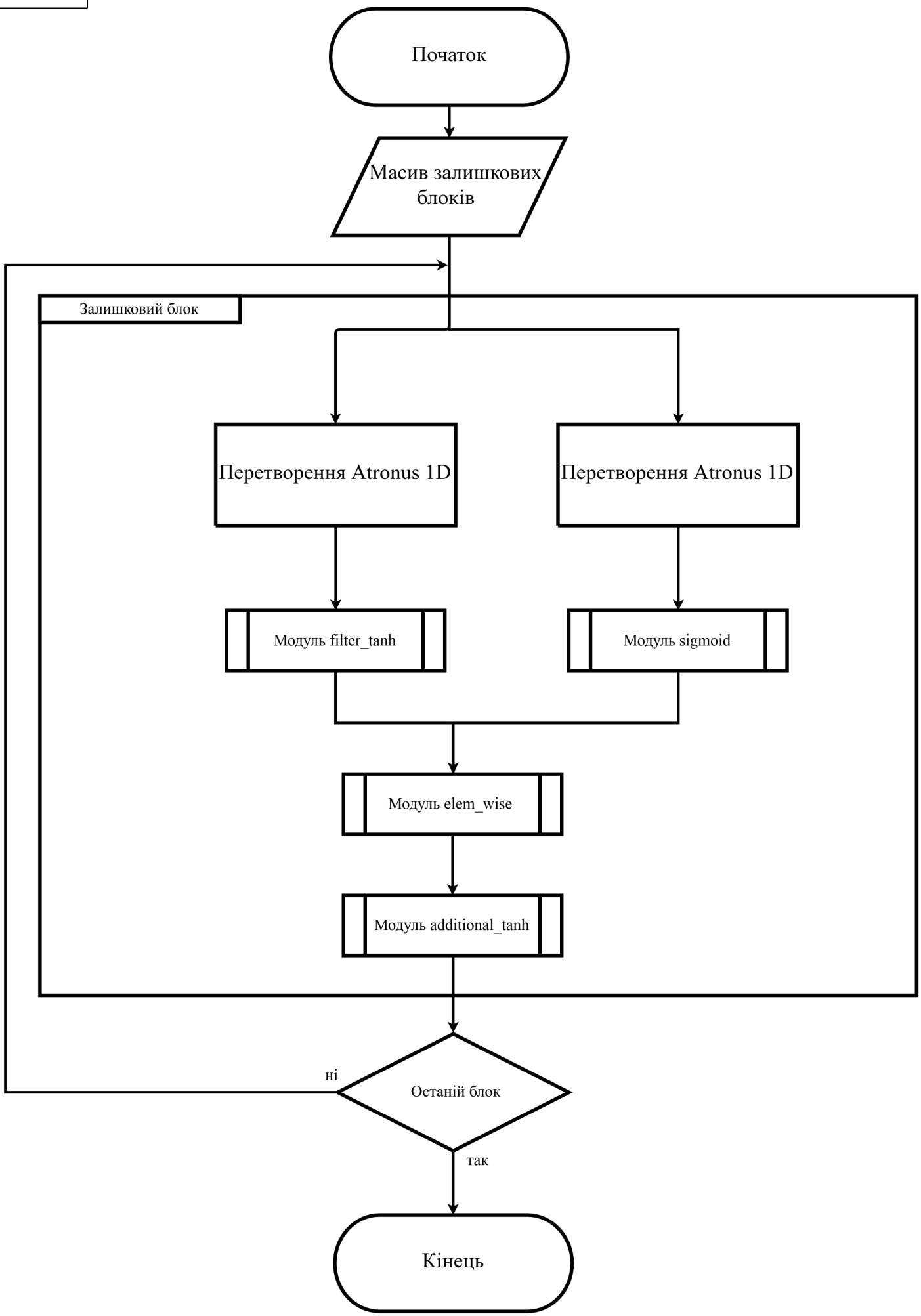
9. S. Mirsamadi and I. Tashev, “Causal speech enhancement combining data-driven learning and suppression rule estimation.,” in Proc. INTERSPEECH, 2016, pp. 2870–2874.
10. Juin-Hwey Chen and Allen Gersho, “Adaptive postfiltering for quality enhancement of coded speech,” IEEE Transactions on Speech and Audio Processing, vol. 3, no. 1, pp. 59–71, 1995.
11. Computer Vision for Music Identification [Електронний ресурс] – Режим доступу до ресурсу: https://www.cs.cmu.edu/~yke/musicretrieval/.
12. K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” in Proc. Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8), 2014.
13. S. Boll, “Suppression of acoustic noise in speech using spectral subtraction,” IEEE Transactions on acoustics, speech, and signal processing, vol. 27, no. 2, pp. 113–120, 1979.

Додаток 1
Копії графічних матеріалів

						ІАЛЦ.045490.005 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата	Фільтрування акустичного сигналу засобами штучного інтелекту	Літ.	Маса	Масштаб
Розроб.		Мартиновець М.В.						
Перев.		Замятін Д.С.			**Високорівневена структура більшості систем фільтрації шуму**	Аркуш 1	Аркушів 1	
Т.контр.								
Н.контр.		Клятченко Я.М.			**Схема структурна**	КПІ ім. Ігоря Сікорського, ФПМ, КВ-51		
Затв.		Тарасенко В.П.						

					ІАЛЦ.045490.006 Д2		
Вин.	Арк.	№ докум.	Підпис	Дата			
Розробив		Мартиновець М.В.			Фільтрування акустичного сигналу засобами штучного інтелекту	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Замятін Д.С.					
					Алгоритм розбору залишкових блоків	НТУУ "КПІ", ФІПМ, КВ-51	
Н. контр.		Клятченко Я.М.					
Затв.		Тарасенко В.П.			**Схема алгоритму**		

					ІАЛЦ.045490.007 Д3	
Змін.	Арк.	№ докум.	Підпис	Дата		
Розробив		Мартиновець М.В.			Фільтрування акустичного сигналу засобами штучного інтелекту	Літ. / Аркуш / Аркушів
Перевірив		Заяєва Д.С.				1 / 1
Н. контр.		Клягченко Я.М.			**Алгоритм розбору залишкових стеків**	НТУУ "КПІ", ФІПМ, КВ-51
Затв.		Тарасенко В.П.			**Схема алгоритму**	



					ІАЛЦ.045490.008 Д4	
Змін.	Арк.	№ докум.	Підпис	Дата		
Розробив		Мартиновська М.В.			Фільтрування акустичного сигналу засобами штучного інтелекту	Літ. / Аркуш / Аркушів
Перевірив		Замятін Д.С.				/ 1 / 1
					Алгоритм розбору залишкових блоків	
Н. контр.		Квятченко Я.М.			Схема алгоритму	НТУУ "КПІ", ФІПМ, КВ-51
Затв.		Тарасенко В.П.				